TÉCNICO LISBOA

Anonymous Authentication With Pseudonyms

Guilherme José Silva Santos

Thesis to obtain the Master of Science Degree in

Telecommunications and Informatics Engineering

Supervisor: Prof. Luís Rodrigues

Examination Committee

Chairperson: Prof. Fernando Manuel Valente Ramos
Supervisor: Prof. Luís Rodrigues
Member of the Committee: Prof. Bernardo Luís Da Silva Ferreira

May 2025

Declaration
I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

This work has been supervised by Prof. Luís Rodrigues in cooperation with Claúdio Correia. I would like to thank them for their guidance and patience.

Secondly, I would like to thank my family, girlfriend and all my friends for safeguarding my mental sanity, over several months thinking about pseudonyms.

Thank you.

Abstract

The problem of anonymous authentication, which is key to preserving the privacy of users, has an increased relevance given the proliferation of applications that use components deployed in the network, that cannot be fully trusted. A possible strategy to preserve user privacy is to implement authentication using pseudonyms. In this thesis, we survey how pseudonyms have been used to protect privacy in different application areas and discuss the advantages and limitations of the different approaches to managing pseudonyms. Based on the limitations of previous work, we design, implement, and evaluate novel techniques to enhance the performance of systems that use pseudonyms.

Keywords

Privacy, Anonymous Authentication; Pseudonyms.

Resumo

O problema da autenticação anónima, uma funcionalidade chave para preservar a privacidade dos utilizadores, tem ganho importância com o aumento de aplicações que recorrem a equipamentos instalados na periferia da rede e nos quais não é possível depositar total confiança. Uma estratégia possível para preservar a privacidade dos utilizadores consiste em usar esquemas de autenticação baseados em pseudónimos. Nesta dissertação, fazemos um levantamento de como os pseudónimos têm sido utilizados para preservar a privacidade em diferentes áreas de aplicação e discutimos as vantagens e limitações das diferentes abordagens para a gestão de pseudónimos. Com base nas limitações dos trabalhos anteriores, propomos, desenvolvemos e avaliamos novas técnicas para melhorar a performance dos sistemas que utilizam pseudónimos.

Palavras Chave

Privacidade; Autenticação Anónima; Pseudónimos.

Contents

List of Figures

List of Tables

Acronyms

BF	Bloom Filter
RL	Revocation List
ZKP	Zero-Knowledge Proof
BSig	Blind Signature
MT	Merkle Tree
HBFA	Hierarchical Bloom Filter Array
RSig	Redactable Signature

1

Introduction

Contents

This thesis addresses the problem of performing anonymous authentication using pseudonyms to ensure the privacy of clients. In particular, we propose novel techniques that improve the efficiency of existing privacy-preserving authentication schemes.

1.1 Motivation

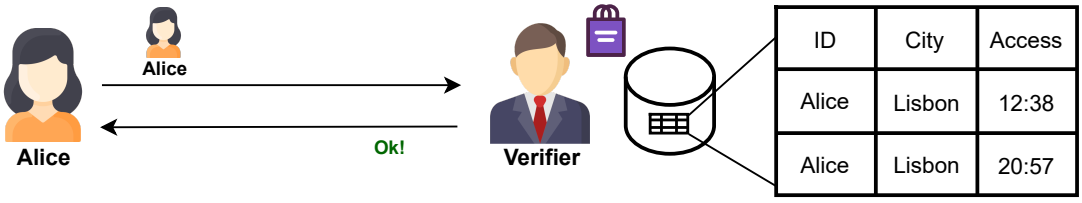


Figure 1.1: Customer profiling through the authentication process.

We motivate our work using smart retail as an example. Smart retail is a general term that captures the use of technologies in retail, in particular digital technologies, to improve customer experience and the efficiency of operations for retailers. Most smart retail applications require customers to authenticate using their identities while shopping. Customers' identities can be used to profile the user behaviour, as shown in Figure 1.1. Customer profiling may bring some advantages to the customer, such as having access to personalized recommendations or promotions but also poses a serious threat to the customer privacy. Information regarding the customer behaviour may disclose personal information, such as health conditions, drinking habits, food regime, etc, that can be used later, against the client interests, for instance, when defining the price of insurance policies, which raises deep ethical concerns.

A possible strategy to preserve customer privacy in smart retail applications is the use of anonymous authentication systems where customers authenticate using different pseudonyms, every time they access the system.

Several anonymous authentication systems with pseudonyms have been proposed for several other applications areas, such as smart vehicles, where similar concerns are raised.

These systems generally involve a central authority, trusted by both clients and application providers, responsible for issuing pseudonyms to clients. Many of these systems offer relevant properties in the context of anonymous authentication with pseudonyms. These properties include, among others: **pseudonym unlinkability**, that guarantee that two different pseudonyms from the same client should be impossible to relate between each other; **revocability**, such that misbehaving customers must be prevented from using previously acquired pseudonyms, usually achieved by revoking the pseudonyms previously issued to the misbehaved client; **backward unlikability**, ensuring that the revocation of pseudonyms does not disclose information about the past use of non-revoked pseudonyms; and **re-**

vocation auditability, that allows clients to verify if a pseudonym has been revoked or not, before using it.

Although many proposed systems offer relevant properties, securing the privacy of clients, these type of systems are inefficient, and suffer from high latencies during the process of authentication, that can take several dozens of seconds, making them unfeasible to be applied in many scenarios, in particular, in smart retail applications.

1.2 Contributions

In this thesis, we survey the main anonymous authentication systems that have been proposed to preserve customers' privacy in multiple applications, that ensure some of the desired privacy properties in the smart retail domain. We propose two techniques that can be applied to anonymous authentication systems, to increase their performace, by reducing the amount of information exchanged during authentication, leveraging Redactable Signatures (RSigs) and Hierarchical Bloom Filter Array (HBFAs) in order to keep the same desired properties when reducing the amount of information exchanged.

1.3 Results

This thesis has produced the following results:

- An implementation of the techniques proposed to reduce the latencies of the authentication, in two different anonymous authentication systems.
- An extensive experimental evaluation of the proposed techniques, covering the latencies of the authentication process and the amount of transferred information during authentication.

1.4 Research History

This work has been performed at INESC-ID, as part of a research effort to improve privacy-preserving authentication in the edge. In my work I have benefited from the collaboration with Cláudio Correia, a PhD student (that has now graduated) working in this field.

Parts of the work described in this thesis have been published as:

- Santos, Guilherme, Cláudio Correia, and Luís Rodrigues. "PickyFilters: Uma Abordagem Prática e Eficiente para Autenticação Anónima com Pseudónimos." Actas do décimo quinto Simpósio de Informática (Inforum), Lisboa, Portugal, Sep. 2024.

This work has been partially funded by the Fundação para a Ciência e a Tecnologia (FCT) via project INESC-ID UIDB/50021/2020 and by the project SmartRetail (funded by IAPMEI with ref. C6632206063-00466847).

1.5 Organization of the Document

The rest of this thesis is organized as follows: Chapter 2 introduces the main concepts relevant to our work. Chapter 3 describes some important anonymous authentication systems proposed for several applications areas, discussing the advantages and disadvantages of each one. Chapter 3 also examines several existing strategies for efficiently transferring Bloom Filters (BFs), which could be valuable for our work. Chapter 4 describes the techniques proposed in this thesis. Chapter 5 shows an evaluation of the proposed techniques. Finally, Chapter 6 concludes the thesis.

2

Background

Contents

In this chapter, we introduce the most relevant concepts for our work.

2.1 Authentication

Authentication is the process of an entity proving to be what it claims to be. This process is usually done when an entity wants to access a protected system or when decides to do a sensitive operation. There are two entities in an authentication process: the entity that tries to authenticate and the verifier that controls this process. The system verifier ensures that the claimed identity is valid by verifying information provided by the entity proving its identity, such as credentials or biometric data. This introduces the issue of a system administrator being able to track and access information about all past authentications and activities of every entity that has authenticated into the system. This is particularly concerning for applications such as smart retail, where customers need to authenticate to acquire goods, and retailers can trace the consumer history. The use of pseudonyms during authentication can circumvent this threat.

2.2 Pseudonyms

Pseudonyms are aliases that clients can use to hide their unique identity. They can be applied to a wide range of privacy-sensitive applications. In information systems, pseudonyms can be used for authentication, replacing the clients' unique identity. Pseudonyms need to be used properly to ensure privacy preservation. In particular, the use of the same pseudonym for multiple authentications raises privacy risks. A system administrator with access to authentication records is able to link authentications that use the same pseudonym, and this may be enough to link the pseudonym to a concrete user. For this reason, users should avoid using the same pseudonym in different authentications. Ideally, users should use a different pseudonym for each authentication.

Changing pseudonyms can only prevent a user's activity from being tracked if the pseudonyms are unlinkable. The property of **unlinkability** ensures that two pseudonyms cannot be connected, even if they are used by the same individual.

2.3 Pseudonym Schemes

In this chapter, we discuss some problems that pseudonym schemes must overcome and some properties that these schemes must provide.

The first challenge a pseudonym scheme must overcome is the need for systems to have enough trust to let customers authenticate with pseudonyms without knowing their real identity. Most of the exist-

ing pseudonym schemes have the pseudonyms used by the clients being issued by a central authority, a pseudonym provider, trusted by both the clients and the verifiers, with the verifiers trusting that the central authority will only provide valid pseudonyms to legitimate clients and with the clients, trusting that the central authority will not disclose the information of mapping between the clients and their pseudonyms.

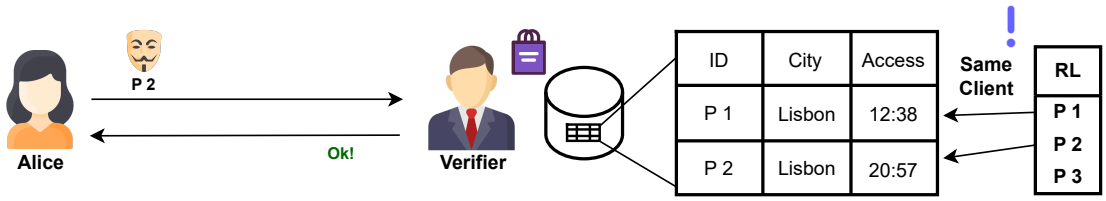


Figure 2.1: Linkability problem when revoking a user. RL is Revocation List. P is Pseudonym.

In many systems, if a client misbehaves or for any other reason, should be removed from the system and should not be allowed to further authenticate in the system. To achieve this requirement, the pseudonym scheme must allow **revocability**, which is the ability to revoke pseudonyms previously issued to a client. Revoking a client in a way that preserves the client's may be challenging. For instance, assume that a pseudonym is revoked by including it in a Revocation List (RL) that is sent to verifiers. If, when a client misbehaves, all pseudonyms of that client are included in the same RL, verifiers may infer that the pseudonyms in the list belong to the same client. If some of these pseudonyms have been used in the past, the verifiers can link the corresponding authentications, as illustrated in Figure 2.1. To avoid this problem, a pseudonym scheme should guarantee **backward unlinkability** and **revocation auditability**.

Revocation auditability states that a client should be able to see his revocation status, before each authentication, avoiding the situation where a client tries to authenticate with a pseudonym that has been revoked without being aware of that fact.

Backward unlinkability states that past authentications must remain private when a client is revoked. Many existing pseudonym systems achieve this property by having the pseudonym providers issuing pseudonyms divided by periods of time, denoted time slots, they should be used in. When a client chooses a pseudonym to use, they use a pseudonym associated with the current time slot. When a client is revoked, only the pseudonyms associated with future time slots are distributed to the verifiers. This solution has one problem, which is, it cannot guarantee perfect **backward unlinkability**. If a client is revoked in a given time slot and the pseudonyms associated to that time slot are distributed to verifiers, if the client has already authenticated in the system in that given time slot, then those authentications' privacy is compromised. If the pseudonyms of the current time slot are not distributed, then the client can authenticate using those pseudonyms until the end of the time slot, when they should be revoked and not authenticating anymore. In order to alleviate this problem, one could consider increasing the

granularity of the time slots, making each time slot really small. This solution has the drawback of increasing substantially, the number of pseudonyms that need to be issued to each client. In systems that use this approach, the number of pseudonyms issued increases linearly with the time slots' granularity, providing the clients with many more pseudonyms than they actually need, increasing the memory needed to store them.

Pseudonym systems should also include the property of **accountability**, which states that it must be possible to trace the actions of a user using a pseudonym back to their real identity. This property is usually assured by maintaining a map between user-pseudonyms in a central trusted authority.

Additionally, a pseudonym scheme must also be efficient, namely, it should be possible to authenticate with low latency, the amount of information distributed in RLs should not be large, the memory space required to execute the protocol should be small, etc.

2.4 Abstractions Used in Pseudonym Based Authentication

In this chapter, we provide some key concepts that are used in existing pseudonym schemes or that we use in our proposed architecture.

2.4.1 Cryptographic Hash Functions

A hash function is a mathematical algorithm that transforms an input (or "message") into a fixed-size string of bytes, typically called a *hash*. A cryptographic hash function extends this by adding essential security properties:

- **One-way:** It is computationally infeasible to reverse-engineer the original input from its hash, ensuring tamper-proof integrity checks.
- **Collision-resistant:** It is highly unlikely to find two different inputs producing the same hash, preventing substitution attacks.
- **Avalanche effect:** A small change on the input drastically alters the output hash, making even minor modifications detectable.

The fact that hash functions are deterministic with these properties make cryptographic hash functions perfect for verifying data integrity.

2.4.2 Asymmetric Cryptography

Asymmetric cryptography is a cryptographic system based on a pair of keys: a private key and a public key. This type of system states that each entity involved must have a pair of keys, publish the public key

and keep the private key, private. Each key works as the decryption key of its pair. Asymmetric cryptography can be used to achieve important properties in information security such as: authenticity, integrity, and non-repudiation. For this reason, asymmetric cryptography is used in most existing pseudonym schemes, using public keys as pseudonyms.

A – Encryption with Asymmetric Cryptography

The encryption is done by encrypting the message with the public key and decrypting it with the private key. Since anyone can use the public key, but only the owner has access to the private key, only the owner can decrypt the message.

B – Digital Signatures with Asymmetric Cryptography

Asymmetric encryption allows the creation of digital signatures that are a cryptographic technique that allows a digital message or document to be verified. The verification of digital signatures ensures that the content has not been altered and prove the identity of the sender. Digital signatures are usually created, simply, by passing the message through a cryptographic hash function and encrypting the resulting digest with the private key. They are sent attached to the messages. The receivers can check a signature by decrypting it with the public key of the sender, passing the message through the same hash function and comparing the results. The integrity property comes from the properties of hash functions, that it is infeasible to find two messages with the same digest. The authenticity property comes from the fact that only the owner of the pair of keys has access to the private key, so that no one else has the ability to create such digital signature.

C – Authentication with Asymmetric Cryptography

As explained before, authentication is the process of an entity proving to be what it claims to be. In the context of using public keys as pseudonyms, this corresponds to an entity proving that it is the owner of a pseudonym (public key) that it is presenting to the system verifier. This is achieved by creating a digital signature and attaching it to the messages sent using the private key associated to the public key in the pseudonym. The receiver, the system verifier, can then verify the digital signature and check if the authenticating entity is who it claims to be.

D – Ensuring Integrity with Asymmetric Cryptography

Integrity ensures that messages have not been altered during their transmission, by man-in-the-middle attacks for instance. This property is vital for sensitive applications, where users' actions must not be changed by an attacker. Asymmetric encryption can also be useful to provide this property. Having public keys as pseudonyms allow the users to use the corresponding private keys in order to create

digital signatures that enable the system to check if the messages sent by the users were altered during their transmission.

E – Ensuring Non-Repudiation with Asymmetric Cryptography

Non-repudiation is achieved when a sender cannot deny the authenticity of the messages sent and operations made. This property is also accomplished using digital signatures. Since digital signatures are only created using the private key and only the owner of the pair of keys has access to it, it is impossible for the senders to deny the authenticity of the messages they sent. This property is closely related to the property of **accountability** of pseudonym schemes, and these two properties together, make it possible to hold an authenticated user accountable for their actions.

2.4.3 Blind Signatures

Blind Signatures (BSigs) are a cryptographic technique that is a form of digital signature in which the content of the digital message or document being signed, is hidden during the process of signing. Different parties are still able to verify the signature when they receive the digital message and its signature, if they have access to the signer's public key. The signer is not privy to what he signed. BSigs can be useful in pseudonym schemes where a signer must not know what they signed. For example, in V-token [1], during the process of pseudonym issuance, the scheme uses BSigs to prevent the central authority of storing pseudonym-identity maps.

2.4.4 Zero-Knowledge Proofs

Zero-Knowledge Proofs (ZKPs) are cryptographic protocols that allow a given entity to prove knowledge about a certain information or statement to another entity wihout revealing the content of that information. ZKPs can be used in pseudonym schemes, during authentications where the clients prove to the system verifiers that none of their previous used pseudonyms are revoked, without revealing which pseudonyms they used before, preventing a system administrator from relating different past authentications. ZKPs have the obvious advantage of privacy and preventing the system verifiers of knowing which pseudonyms, clients used before. On the other side, ZKPs are computationally expensive.

2.4.5 Accumulators

Dynamic accumulators [2] are a constant-size cryptographic construct that represents a set membership. It is possible to add or remove elements from the accumulator. Besides that, accumulators allow anyone to prove in zero knowledge that some element is in that accumulator if and only if, the element is in the accumulator. It is possible to use these accumulators as a ”whitelist” of valid pseudonyms. Universal

accumulators [3], allow users to prove instead that an element is not in the accumulator. Universal accumulators can be used as a ”blacklist” of revoked pseudonyms.

2.5 Bloom Filters

BFs [4] are a probabilistic structure designed to store members of a set and to determine if a given element is member. In pseudonym systems, verifiers often use BFs for the storage of the revoked pseudonyms. BFs are very eficient and have constant computation costs $O(1)$ for storing and searching elements. The filter is made up of N addressable bits, with addresses from 0 to N-1, being N its size. Besides, its size, a filter also has another paramenter which is the number of hash functions, k. When an element is to be inserted in the filter, it is hash coded using the hash functions, obtaining k hash codes, that will be used as addresses inside the filter. After that, the bits of the filter with that addresses, are set to 1. To test whether an element belongs to the filter, the element is hash coded using the hash functions obtaining k addresses, if and only if the bits in the filter with that addresses are all set to 1, the element is considered to be in the filter. When searching for an element, BFs may indicate that the element is in the filter when it is not, allowing false positives to happen. On the contrary, false negatives are not possible. The rate of false positives depends on the size of the filter, N, the number of elements inserted, m, and the number of hash functions used, k. The rate of false positives can be calculated using the following formula:

$$P = (1 - (1 - \frac{1}{m})^{kn})^k \tag{2.1}$$

BFs also provide an efficient operation to merge different filters of the same size into one. If the BFs have the same characteristics, as hash functions and size, it is possible to merge different filters by simply performing a bitwise OR operation.

2.6 Hierarchical Bloom filter Arrays

HBFAs [5] are a data structure based on BFs that aim to increase the scalability of the original BFs. These filters use a hierarchical structure composed of multiple BFs of different sizes (with the filters in the higher positions of the hierarchy being smaller than the filters in the lower positions), where the registration/test of presence is done by utilizing multiple filters. Typically, a membership test in the data structure requires performing sequential presence tests across the various filters until a conclusion is reached.

2.7 Redactable Signatures

RSigs [6] enable the sending of a digitally signed message, with the ability to delete certain parts of it, while still allowing the recipient to verify the security properties of the message, such as integrity and authenticity. The process of creating a redactable signature begins by dividing the message into removable parts. A Merkle Tree (MT) is constructed by associating each part of the message with a leaf node of the tree, calculating its cryptographic hash, and then recursively building the tree up to the root node, where each node's value is the hash of the concatenation of its child nodes' hashes. After that, the sender digitally signs the root node.

When sending the message, the sender deletes the parts they wish to omit and sends the remaining parts to the recipient. Along with the message, they also send the hashes of the leaf nodes corresponding to the deleted parts. These hashes can be compressed using the tree structure by sending the hashes of internal nodes. The recipient uses the received information to reconstruct the hash of the root node and verify the sender's digital signature on that node.

RSigs also incorporate a random component during the signature creation process, which prevents the recipient from recovering the deleted parts through brute force attacks.

Summary

In this chapter, we provided the main background for our work. In the next chapter, we will describe how these abstractions have been applied to implement pseudonym-based authentication protocols, detailing the design and functionality of these protocols in addressing privacy and security challenges in authentication systems.

3

Related Work

Contents

This chapter is divided in two sub chapters, where we discuss some relevant work in the field. In Chapter 3.1, we explain some existing pseudonym systems detailing their characteristics, which properties they achieve and how, while discussing their flaws. In the Chapter 3.2, we discuss two different strategies to transfer BFs efficiently, that could be used to transfer RLs if they are implemented using BFs.

3.1 Anonymous Authentication Systems with Pseudonyms

We start by explaining the general structure that almost every pseudonym system follows. Generally, this kind of systems are usually composed by 3 entities: 1) the clients, 2) the verifiers that control the authentication process and, 3) a central authority trusted by both clients and verifiers which is a pseudonym provider, that provides clients with enough pseudonyms for their authentications, keeps a map between users' real identities and the pseudonyms issued, and handles clients' revocations, distributing the revoked pseudonyms to the system verifiers using RLs, as shown in Figure 3.1.

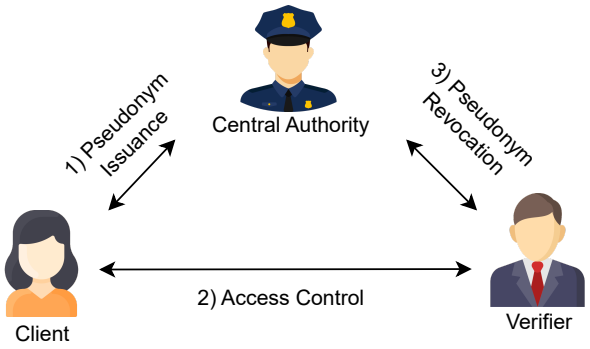


Figure 3.1: Anonymous Authentication Systems Overview.

The pseudonyms issued have normally an asymmetric key pair associated and are made up of the public key and the pseudonym provider's signature. When a client is authenticating in the system, he uses a pseudonym as his digital identity and sends a different one to the system verifier in each authentication. The purpose of the pseudonym's signature is to prove to the verifier the authenticity and integrity of the pseudonym showing that the pseudonym was, indeed, created by the pseudonym provider. This scheme takes advantage of the asymmetric encryption properties to prove that the client is the real owner of a given pseudonym, by using the associated private key. So, the messages exchanged between the system verifier and a client are usually appended with the pseudonym and a signature made by the client using the private key associated to that pseudonym.

These systems also follow similar steps, as shown in Figure 3.1. These steps are usually divided in 3

phases: 1) Pseudonym Issuance, 2) Access Control and, 3) Pseudonym Revocation. Whenever a client joins the system, must start by asking the pseudonym provider for pseudonyms for his authentications. After acquiring his pseudonyms, a client is ready to authenticate in the system with a verifier. The authentication is divided in three phases: the verifier sends to the client, information about his revocation status, the client checks this information and if he is not revoked, sends a pseudonym to the verifier, the verifier checks the validity and authenticity of this pseudonym and if everything is correct the user accesses the system. If an authenticated client misbehaves, the system generates a complaint, inserting the pseudonym the client used, and sends it to the pseudonym provider. The pseudonym provider links the pseudonym inserted in the complaint to the real identity of a client and then revokes all the pseudonyms issued before to that client and distributes them in a RL to the system verifiers.

This is just a general approach followed by most systems, but some other systems may present some radical approaches or simply do not implement some steps of this approach. In the next chapters we take a deeper look at existing pseudonym systems and discuss them.

3.1.1 Approaches based on Epochs and Time Slots

As mentioned before, some pseudonym systems have a similar approach to achieve *backward unlinkability*, Figure 3.2. They divide the time into large periods of time denoted as epochs and these are divided into smaller intervals denoted as time slots. The pseudonyms issued to a given client are divided into groups and each group is associated to one of the slots and the pseudonyms cannot be used outside of their slot. When revoking a client, only pseudonyms associated to future slots are inserted in the RL and distributed to the verifiers.

As explained in chapter 2.3, this type of approach does not achieve perfect **backward unlinkability**, due to the fact that revoking a client in a given time slot has the problem of exposing the user's authentications in that given time slot or the problem of letting a misbehaving user continue to authenticate in the system until the end of that time slot. This can be alleviated augmenting the granularity of the time slot which would increase linearly the number of pseudonyms issued. This type of schemes have the challenge of finding a good trade-off between the size of the slots, that should be as small as possible, and the number of pseudonyms issued.

Another disadvantage is the fact that this approach forces to provide the client with pseudonyms for all the time slots that the client can possibly authenticate in the system, even if the client only authenticates in the system in one time slot, providing the client many more pseudonyms than he actually needs. An ideal solution would be to provide the client as many pseudonyms as the number of authentications he performs.

Next, we present some pseudonym systems that use this notion of epochs and time slots.

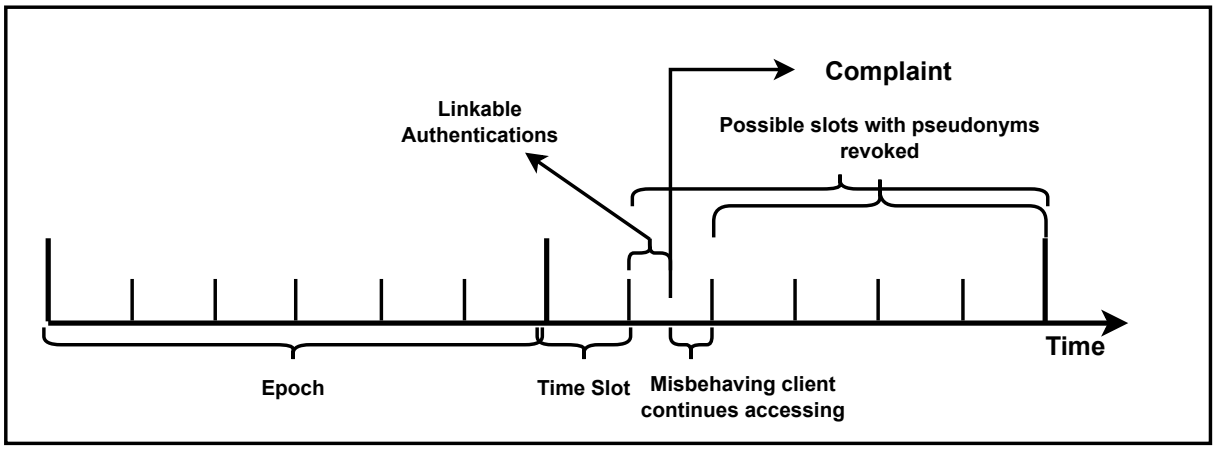


Figure 3.2: Epochs and Time Slots Approach.

3.1.1.A Haas et al.

Haas *et al*. [7] proposes a lightweight mechanism to revoke pseudonyms in an anonymous scheme developed for the authentication of messages exchanged between vehicles in VANETs.

The pseudonyms are issued by a central authority, being also responsible for revoking the pseudonyms of misbehaving vehicles. This work uses the concept of epochs, a large period of time for which the pseudonyms are issued, and each epoch is divided in multiple time slots. The pseudonyms that are issued for a given vehicle and epochs are divided in groups, and each group is assigned a time slot where the pseudonyms of the group must be used. The system achieves partly the property of **backward unlinkability** keeping the past authentications of the client private, by revoking only pseudonyms of future time slots.

Pseudonym Issuance For the Pseudonym Issuance phase, this work developed an algorithm for the creation of the pseudonyms to achieve an efficient way of generating and revoking the pseudonyms. Whenever the certificate authority receives a registration request from a client, it creates a nonce which will be used to create a hash chain as long as the number of time slots in the epoch. Each element of the chain is associated to a time slot and is created hashing the previous element, being the first one obtained hashing the nonce created for that client, as illustrated in Figure 3.3. After generating the chain, the certificate authority generates the groups of pseudonyms to be sent to the client. For each time slot, it generates a sequence of values from 1 to R, being R the size of the pseudonym group, and encrypts each of these values using the associated element of the hash chain as the key of a cipher. Each of these ciphertexts is the identifier of a pseudonym and is inserted on it. The certificate authority generates pairs of asymmetric keys, $\langle K^+, K^- \rangle$, and include one of these pairs in each pseudonym, finishing their creation by adding its digital signature generated from the identifier and the public key of the pseudonym, with its private key, K_{CA}^-. After all these steps, the certificate authority replies to the

client sending the pseudonyms it just created. Being each pseudonym $p = \langle id, K_p^+, \{id, K_p^+\}^{K_{AC}^-} \rangle$

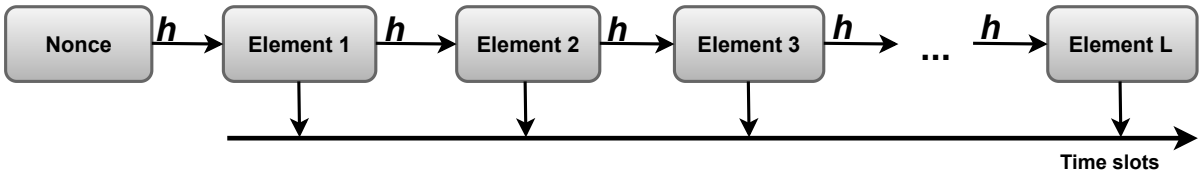


Figure 3.3: Hash Chain.

Access Control In order to authenticate himself, a client sends his message along with a pseudonym of a group associated with the current time slot. Whenever a client receives a message, it checks the pseudonym that comes with it, verifying the digital signature and checking that its identifier is not in its ”blacklist” of revoked pseudonyms. This work also includes a way to improve the efficiency of the memory necessary to store the revoked clients' information and the look-ups in the RL, which is the usage of BFs to store the revoked pseudonyms.

Pseudonym Revocation To revoke a client, the certificate authority publishes and distributes the element of the hash chain associated to the current time slot. With this information, all the vehicles perform the same steps used in the creation of the pseudonyms and obtain the identifiers of the revoked client's pseudonyms for the current time slot and the future ones. The privacy of the revoked client's past authentications is assured by the irreversibility property of hash functions which guarantees that given an element of the chain, it is not possible to calculate the previous elements.

This approach has several advantages such as the minimal information necessary for the certificate authority to store about each customer which is just the nonce associated to the client's registration, the amount of information necessary for the certificate authority to revoke a client which is just the element of the hash chain associated with the time slot when the client was revoked.

On the other hand, this scheme is based in epochs and time slots, which does not allow to achieve revocation with perfect *backward unlinkability*. It is possible to argue that it would be possible to shorten the time slots in order to make it infeasible to revoke a client in a time slot that the client has already authenticated in, but this would increase the number of pseudonyms necessary to be issued to each client linearly with the granularity of the time slots, leading to a huge number of pseudonyms issued and not used by the clients. It must be found a good trade-off between the size of time slots and number of pseudonyms issued.

3.1.1.B EDGAR

EDGAR [8] introduces a new class of pseudonyms known as ”Range-Revocable Pseudonyms” that can be revoked for any time range within its original period. This scheme, also constructed with the concept of epochs and time slots, tries to overcome some flaws of Haas *et al.* [7]. This scheme has a pseudonym

manager issuing clients' pseudonyms, and each pseudonym has an associated key pair, used by the clients to authenticate themselves. The proposed scheme solves the problem of issuing many more pseudonyms than the client actually needs, permitting to issue only as many pseudonyms as the number of authentications done by the client. This is achieved because the information distributed when revoking a given pseudonym cannot be linked with its usage outside of the revoked time range. This paper also improves the perfect **backward unlinkability** problem, but does not achieve it completely.

This work uses the concepts of pseudonyms and capabilities. In the beginning of each epoch, a tree is constructed with as many leaf nodes as the number of slots in an epoch, and each slot is associated with a leaf node, as shown in Figure 3.4. Each node is assigned a label that the clients later use to create capabilities and authenticate in the system.

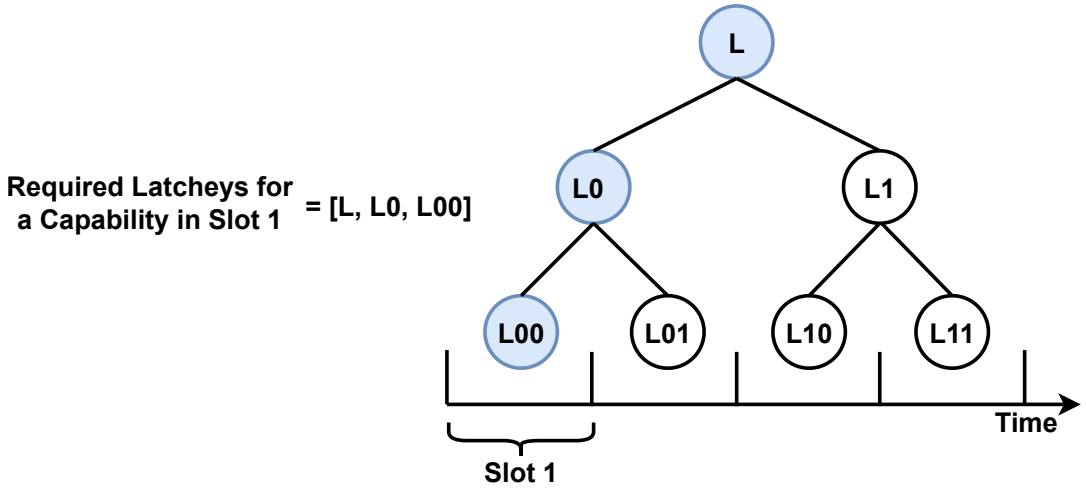


Figure 3.4: Latchkeys and Capabilities.

Pseudonym Issuance When joining the system, the client asks the pseudonym provider for pseudonyms and receives a bunch of these, valid for the current epoch, each with an associated asymmetric key pair. The pseudonym manager signs the pseudonym to ensure its integrity and authenticity.

Access Control Capabilities are used for authenticating with the system verifiers. When authenticating, the clients must generate a capability using one of their pseudonyms, for the specific time slot they are authenticating in. To generate a capability, the user signs the label of each node that form the path between the root node and the leaf node associated to that specific time slot, using the chosen pseudonym's private key. These signatures are denoted latchkeys and a capability is defined as the set of these latchkeys, the public key and the signature of the pseudonym provider of the pseudonym which originated this capability. After creating the capability, the client sends the capability and the pseudonym to the system verifier. When a verifier receives a capability, verifies the pseudonym manager's signature, with the public key, verifies if the latchkeys were originated using the private key of the pseudonym and

if they were originated from the labels of the tree's nodes associated with that time slot. Then it checks, if any of the latchkeys is in the RL. If none of them is, the client is authenticated.

Pseudonym Revocation To revoke a pseudonym, the pseudonym manager must revoke the capabilities of that pseudonym for future time slots. A capability is valid only when none of its latchkeys is revoked. In order to revoke a capability associated to a given time slot, the pseudonym manager can revoke the latchkey of the leaf node. If the pseudonym manager wants to revoke capabilities of several time slots, it can revoke inner nodes' latchkeys of the tree, revoking all the capabilities associated with leaf nodes of the sub-tree that has those inner nodes as root. The pseudonym provider then calculates the latchkeys for the pseudonym being revoked, using its private key, and distributes them to the verifiers. The capabilities generated from the same pseudonym have intersecting information, so the user must never use the same pseudonym twice, as the two authentications could be linked. When revoking a client, the pseudonym manager must not revoke latchkeys which may have already been used in capabilities of previous time slots.

This work allows to increase the granularity of the time slots, without increasing the number of pseudonyms used, but at the cost of incresing the revocation information to be distributed to the verifiers, because all the future possible capabilities have to be revoked and the number of capabilities increase with the granularity. Despite this increase, the tree system allows to alleviate this problem, revoking latchkeys of inner nodes.

This aproach has the advantage of the amount of information necessary for the pseudonym manager to store about each client, which is just the clients' identifiers. This work uses a pseudorandom function with a seed composed of the client identifier, epoch and time slot, in order to create the pseudonyms for a given client. With these client identifiers, when revoking a client, the pseudonym manager can reconstruct the seeds, generate the pseudonyms again and distribute them to the verifiers.

When compared with Haas *et al*. [7], this work allows to increase the granularity of time slots, in order to attenuate the perfect *backward unlinkability* problem, at the expense of having much bigger RLs.

3.1.1.C NYMBLE

This work [9] designed a pseudonym scheme for anonymizing networks such as Tor. Clients use this kind of networks, when accessing websites, to hide their IP addresses through a set of nodes belonging to the network. Since website administrators can't block a misbehaving user by blocking his IP address, they usually block the IP address of the exit node, blocking not only the misbehaving user but also all the other legitimate users. The presented scheme tries to solve this problem.

The scheme is composed by four entities: user, pseudonym manager, nymble manager and server. The users start by asking the pseudonym manager for pseudonyms and then use those pseudonyms to get a credential from the nymble manager. With this credential the user is able to authenticate in a given

server keeping their privacy. The detailed process is explained below.

Pseudonym Issuance Like in the previous systems, time in this system is divided into epochs, and each epoch is further divided into slots. When a client joins the system, they begin by requesting a pseudonym from the pseudonym manager.

After obtaining the pseudonym, the client can access servers by using this pseudonym to acquire a credential from the nymble manager. Each credential is specific to a single server; thus, if the client intends to access multiple servers, they must obtain a separate credential for each server.

A credential consists of a sequence of nymble tickets, one for each slot of the epoch. Each nymble ticket is valid only for the corresponding slot and must be used within it. When authenticating with a server, the client presents the appropriate nymble ticket.

In the first step, the pseudonym manager processes a request from the client and returns a pseudonym valid for the current epoch. This pseudonym consists of two Message Authentication Codes (MACs):

1. The first MAC is computed using the client's identifier, the epoch, and a secret key known only to the pseudonym manager.

2. The second MAC is computed using the first MAC, the epoch, and a secret key shared between the pseudonym manager and the nymble manager.

When a client presents their pseudonym to the nymble manager, the nymble manager begins by verifying that the pseudonym is legitimate by checking the second MAC. If the MAC is valid, the nymble manager proceeds to create the credential.

First, it generates a seed associated with the pseudonym presented by the client. Using this seed, the nymble manager creates a hash chain with a length equal to the number of slots per epoch plus one, utilizing an irreversible hash function f. Each element of this chain, except for the initial element, is associated with a specific slot. The second element corresponds to the first slot, the third element to the second slot, and so on.

Using this hash chain, the nymble manager creates the credential. First, it applies another irreversible hash function g to the first element of the chain to produce the credential identifier, denoted as nymble*. Then, the manager iterates through the hash chain, passing each element through the hash function g to create the identifiers for the nymble tickets.

To construct each nymble ticket, the nymble manager generates two MACs and a ciphertext:

- The first MAC is computed using the server's identifier, the slot associated with the nymble ticket, the epoch, the nymble ticket's identifier, and a secret key known only by the nymble manager.

- The second MAC is computed using all the information used in the first MAC, along with the first MAC itself, but employing a secret key shared between the server where the client will use this credential and the nymble manager.

- The ciphertext is created by encrypting the concatenation of the credential identifier (nymble*) with the element of the hash chain associated with the slot for which the nymble ticket is being created.

Each nymble ticket is represented as a tuple consisting of the slot in which it must be used, the nymble ticket's identifier, the two MACs, and the ciphertext.

After completing the construction of all the tickets, they are grouped together and inserted into the credential, which is then sent to the client.

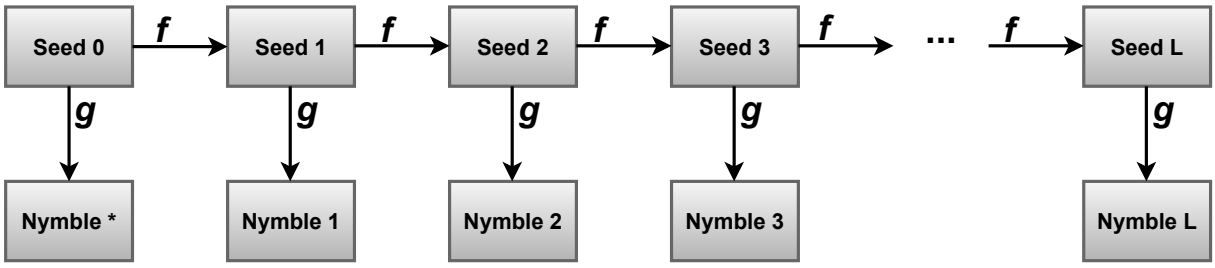


Figure 3.5: Creation of Nymble Tickets.

Access Control With the credential, a client can authenticate in a server, sending the ticket associated with the current slot. When the server receives a ticket, it verifies the ticket, checking if the ticket was indeed created by the nymble manager using the second MAC of the ticket and checking if the nymble ticket is not revoked, by verifying if it is present in the server's "blacklist". The server allows the authentication if this verification is successful.

As mentioned before, this scheme provides the clients with **revocation auditability**, by allowing them to verify their revocation status before authenticating in the system. In the beginning of each authentication, the server sends its "blacklist" to the user and the user only proceeds with the authentication if his revocation status is negative.

This method could raise some problems, such as, the server sending an outdated "blacklist" to the client. The authors present a solution, that allows the clients to check the integrity and freshness of the "blacklist". This is accomplished by having the nymble manager signing the "blacklist" for every server in the beginning of each time slot. When a client receives the "blacklist" from a given server, he checks the nymble manager's signature and verifies that the "blacklist" corresponds to that time slot and it is complete.

Pseudonym Revocation When a client misbehaves at a given server, the server has the capability of revoking that client by generating a complaint and sending it to the nymble manager, including the nymble ticket that was used by the client when connecting. Receiving a complaint, the nymble manager checks the authenticity of that nymble ticket using the first MAC inserted in it, decrypts the ciphertext and gets the nymble* which is the identifier of the credential where that nymble ticket* is inserted. The nymble manager returns the element of the chain created when creating that credential, associated to

the next time slot. With this element, the server is capable of generating the nymble ticket's identifiers for the future time slots and revoke them, inserting them in its "blacklist".

Comparing this system with the previous ones, Nymble, obviously, provides **revocation auditability** which is the biggest improvement. Besides that, the pseudonym issuing is divided by 2 central authorities and none of them, has the ability to resolve, nymble tickets or pseudonyms to a real identity, which increases the privacy of user that is not exposed to a single central authority. Despite of these improvements, the nymble manager keeps some of the flaws of the previous works. It is a scheme based in epochs and time slots, which does not allow to achieve revocation with perfect **backward unlinkability**. This problem is aggravated by the mechanism used to achieve **revocation auditability**. The need of having the nymble manager signing the "blacklists" in the beginning of each time slot, forcing the nymble manager to sign all the "blacklists" in the beginning of each slot, forces the nymble manager to store these "blacklists", or to generate them in the beginning of each slot, increasing the memory space used or the computation necessary. This problem, makes impossible to reduce the time slots to alleviate the perfect revocation problem, since, once reducing them, this signing process would be repeated more often.

3.1.1.D IFAL

IFAL [10] proposes a pseudonym scheme for the authentication of messages exchanged between vehicles in VANETs. This work presents an original approach. While other existing pseudonym schemes issue pseudonyms to clients, that are able to authenticate in the system right after receiving the pseudonyms, IFAL has the approach of pre-issuing the pseudonyms to clients and activate them later by sending activation codes. These activation codes allow the vehicle, to generate the private keys associated to the pseudonyms.

This work has the advantage of not needing to revoke the pseudonyms and distribute them to the verifiers in RLs, unlike other systems. Instead of revocating the pseudonyms of a misbehaving client, IFAL stops sending activation codes to a revoked client and this client runs out of valid pseudonyms and is prevented from authenticating again. This system has the following entities: Enrollment Authority, EA, responsible for the vehicles' registration in the system, Authorization Authority, AA, responsible for issuing the vehicles' pseudonyms.

Pseudonym Issuance In this model, the vehicle starts by generating a registration request and sending it to the Enrollment Authority. The EA assigns a unique uid to the vehicle, signs the request together with the uid and sends it back to the vehicle. Next, the vehicle sends the information, it received from the EA, to the AA in a request for pseudonyms. The AA checks the information received and generates a certificate file with multiple pseudonyms, and the activation codes for the pseudonyms. The certificate file is returned to the vehicle and the activation codes are stored together with the vehicle's

uid.

Time is divided in epochs and the pseudonyms inserted in the certificate file are divided into groups and each group is associated to a given epoch. Pseudonyms cannot be used outside of the epoch they are associated to.

Activation Phase In the activation phase, the AA distributes activation codes to vehicles and an activation code permits to generate the private keys for all pseudonyms associated to a given epoch. The AA maintains in its storage, a mapping between $uids$ and activation codes generated for each uid. The AA iterates through this list and sends the activation code for the next epoch to the client, through the EA. The EA maintains in its storage, a mapping between uids and the canonical identities of the vehicles. The EA receives the activation codes from the AA and sends them to the vehicle corresponding the uid indicated by the AA.

When a vehicle receives an activation code, it calculates the private keys for the pseudonyms of that epoch, and once it gets the keys, the vehicle is ready to authenticate its messages.

Pseudonym Revocation In order to revoke a vehicle, there are two possible ways. The first way is that the EA receives a request to revoke a vehicle with its canonical identity. In this case, the EA gets the vehicle's uid and sends it to the AA informing the AA to stop issuing activation codes to the vehicle with this uid. In the second mechanism, the AA receives a complaint about a misbehaving vehicle with the pseudonym used during the authentication. The AA uses this pseudonym to recover the uid of the vehicle and stops sending activation codes for this vehicle.

This system does not achieve immediate revocation, as it is necessary to wait for the beginning of the new epoch for the revoked client to run out of pseudonyms. This is as problematic as the size of the epochs and reducing them would not be a solution, since it would increase the number of pseudonyms used and the number of activation codes sent.

This work has a good improvement which is the replacement of distribution of RLs by the the distribution of activation codes. The distribution of activation codes would be a lot more efficient since the activation codes are much smaller than the RLs. This would be good in a scenario where the number of verifiers are similar to the number of clients as a VANET, where all vehicles work as verifier and client, but not in a scenario where there are very few verifiers comparing to the number of clients as in smart retail applications. The last scenario, would be a scenario where the overhead caused by RLs distribution would not be so high and there would be much more activation codes circulating if we used this approach, than RLs if we used the previous approaches. Even if the activation codes are more efficient, this would be a downgrade overall.

3.1.2 Other Approaches

Some schemes present different strategies, other than the notion of epochs and time slots. This is the case of V-tokens [1], PEREA [11] and Privacy Keeper [12].

3.1.2.A V-token.

V-token [1] is a pseudonym scheme developed for VANETs where vehicles need to authenticate their messages. While most of other similar works rely the resolvability of pseudonyms to issuance authorities that assure this by storing pseudonym-identity mappings, this work argues that these mappings are too privacy sensitive and present a pseudonym issuance protocol that satisfies the resolvability of pseudonyms while preventing the issuance authorities from storing pseudonym-identity mappings.

There are 4 entities in this scheme: vehicles, certificate authorities, CA, pseudonym providers, PP and resolution authorities, RA. When a vehicle joins the system, it is given a long term identifier, id-V, which is a certificate and the corresponding pair of keys, by a certificate authority. A vehicle can obtain pseudonyms from pseudonym providers and before the pseudonyms are issued, it is verified if the vehicle has been revoked. The resolution authorities take part in the resolution process. This work achieves its purpose of preventing the issuance authorities from storing pseudonym-identity mappings by inserting the resolution information directly in the pseudonym. This piece of information is known as V-token.

The protocol of issuance is divided in two phases: **Authentication Phase** and **Acquisition Phase**.

Authentication Phase In the Authentication Phase the vehicle starts by sending the certificate authority a request in order to receive V-tokens, including his id-V and a signature using his long term certificate. The CA checks the vehicle is not revoked and sends back an id which is the concatenation between id-V and id-CA, being id-CA the identifier of this CA, the public key of the resolution authorities and requests N commitments. After this, the vehicle creates N V-tokens, by generating N unique random number for each and appending them to the id it just received, and encrypting this concatenation with the public key of the resolution authority. After generating the V-tokens, the vehicle chooses N random distinct blinding factors and blinds each commitment using one of these factors. The client stores the used factors and the unique random numbers used when creating the V-tokens and sends the set of blinded V-tokens to the CA. The vehicle must prove probabilistically that it has created the V-tokens correctly, using the id provided by the CA. This is achieved by having the CA requesting some blinded V-tokens for the vehicle to reveal. The client then sends the blinding factors and the unique random numbers used when created each of the V-tokens that he must reveal now. With this information the CA, unblinds the V-tokens and check if they were well constructed. At the end of this phase, the CA signs the remaining blinded V-tokens and sends these signatures to the vehicle. The certificate authorities are not able to understand which V-tokens they signed because of the BSigs, so, they won't be able to trace

the pseudonyms to a given vehicle.

Acquisition Phase During the Acquisition phase, the vehicle gets pseudonyms from a pseudonym provider. Once a vehicle has in its possession signed V-tokens, it contacts a pseudonym provider to get a pseudonym for each V-token. To generate a pseudonym, the vehicle starts by generating a key pair and generates a request to the pseudonym provider, that includes the public key generated and one V-token and signs the request with the private key, showing its ownership of the key pair. When the pseudonym provider receives such request, it starts by verifying the vehicle's signature and checks the validity of the V-token, by checking the CA's signature using the CA's public key. If everything is good, the PP creates a pseudonym, inserting the public key and the V-token received in the request, signs it and sends it to the vehicle.

The process of mapping a pseudonym to a real identity, corresponds to decrypt the V-token inserted in the pseudonym with the private key of the Resolution Authority, in order, to obtain the id-V and the id-CA. With this information would be possible to map the id-V of the vehicle to a real identity, by contacting the Certificate Authority that registered the vehicle and created its V-tokens.

This work does not provide any process for the revocation of already issued V-tokens.

The main purpose of not having a central authority storing pseudonym-identity mappings is achieved. Another good advantage of this scheme, is that the vehicles only need to generate as many pseudonyms as the number of authentications they perform, unlike other schemes. On the other side, the process of generating V-tokens and pseudonyms is complex and perhaps, it creates a little overhead on the vehicles' system. Also, it would be possible for a lucky vehicle to generate invalid V-tokens during the Authentication Phase. For a revoked vehicle, it would be necessary to wait until the client runs out of valid V-tokens before he loses the ability to authenticate in the system, which is also a disadvantage.

3.1.2.B PEREA

PEREA [11] presents a pseudonym scheme designed for websites, that provides perfect **backward unlinkability** and **revocation auditability**, without the use of third trusted parties. It uses an accumulator as "blacklist" for revoked pseudonyms, kept by the website system. The property of **revocation auditability** is achieved by having the website sending the "blacklist" to the user in the beginning of each authentication. The property of **backward unlinkability** is achieved by never letting the website system know a set of pseudonyms used by a client.

Pseudonym Issuance The clients start by authenticating themselves with the website system and obtain a credential that allows them to generate future pseudonyms to authenticate in the system. Whenever a client authenticates, the client generates a new "ticket" himself and authenticates in the system with that "ticket" that serves as a pseudonym.

Access Control In order to authenticate successfully, the client also needs to prove that none of

his k last "tickets" is in the "blacklist" of the website system. The key feature of this work, is the use of ZKPs to generate a non-membership proof for all the k last "tickets" the client used, proving that the k last "tickets" are not inserted in the accumulator. The use of ZKPs prevents the website system of relating the set of the last k "tickets" to the user and gain information about his last k authentications. Furthermore, the client also sends another ZKPs in order to prove that he didn't replace any of the last legitimate k "tickets". The variable "k" is application dependent, it must be a reasonable value slightly higher than the number of expected authentications before the website system recognizes a misbehavior.

This pseudonym scheme has the advantages of providing perfect **backward unlinkability** and **revocation auditability**. The number of pseudonyms created for each user is equal to the number of authentications performed. On the other side, this scheme uses ZKPs to a very large extent, techniques that are very resource intensive. Moreover, the number of ZKPs used in each authentication depends on the value of k. This scheme may be impractical for applications with a large "k". This pseudonym scheme also has the disadvantage that if a user is able to authenticate more than k times, after he has misbehaved, and before the system notices this misbehavior, he will continue being able to authenticate in the system, even after the system has "blacklisted" his pseudonym used during the misbehavior. The authors argue that it would be possible to limit the clients' authentications. This measure would be a disadvantage by itself and would be impossible to implement in a physical system, like smart retail applications.

3.1.2.C Privacy Keeper

This work [12], illustrated in Figure 3.6, presents a totally different approach to achieve perfect **backward unlinkability** when revoking a client, assuring at the same time **revocation auditability** to the clients. This work presents an approach where the client sends a pseudonym and a proof of non revocation in order to authenticate in the system. Instead of revoking the pseudonyms and distributing them directly to the verifiers, the central authority distributes the proofs used by the clients to authenticate, in RLs. The fact that the verifiers never get to know which pseudonyms are revoked, makes it impossible for them to associate a bunch of pseudonyms to the same user as in other anonymous schemes. Also, this work is not based in epochs and time slots which permits the central authority to provide the client with only as many pseudonyms as authentications done by the client.

Pseudonym Issuance The pseudonyms are created by the central authority and sent to the clients. The pseudonyms have an asymmetric key associated and are formed by the public key and a digital signature made by the central authority to prove the pseudonym's validity. The non revocation proofs are digital signatures of a seed created with the private key associated to a pseudonym.

Access Control When revoking a misbehaving client, the central authority starts by creating a new RL, a new seed associated to that RL, generated randomly, and then generates the non-revocation

proofs for the pseudonyms of the client revoked using the seed it just created and the proofs of non revocation for all the previous revoked pseudonyms. The central authority keeps track of pseudonyms issued to each user. Finally, gets all the proofs together in the RL, signs the concatenation between the list and its seed and distributes these elements to the verifiers.

Pseudonym Revocation When a client is authenticating, the verifier sends him the last RL it received from the central authority, the seed and the signature of the central authority. The client verifies the signature, chooses one of his pseudonyms and calculates the non-revocation proof with the seed received for that pseudonym. If this proof is not in the RL received, then it means that the client is not revoked and if he is not, he finishes his authentication sending the pseudonym and proof to the verifier, otherwise, he aborts immediately. The verifier, checks the authenticity of the proof using the pseudonym's public key and checks if the proof is in the RL, if it is not, the client is authenticated.

The property of **revocation auditability** is achieved because the client can see if is revoked before sending any information to the verifier. The perfect **backward unlinkability** is achieved because the verifiers never get to know the revoked pseudonyms, only their non revocation proofs. The verifier has the ability of associating a bunch of proofs to the same client, but the client will never send one of this proofs because he checks if his proofs are in the RL sent by the verifier before sending them. Even if the verifier sends an outdated RL to a revoked client, of a time where he was not revoked yet, the client will calculate the proofs based on the seed associated to this RL and they will be different of his proofs that are present in the updated RL. This is assured by always having a different seed for each RL. Of course, the verifier can't also forge RLs because their integrity and authenticity is protected by the digital signature of the central authority.

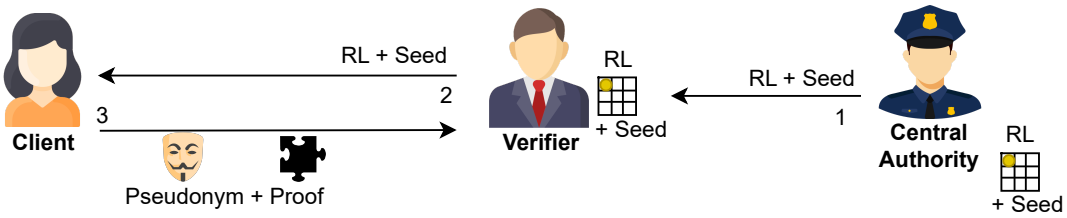


Figure 3.6: Authentication in Privacy Keeper. RL is Revocation List.

This work has 2 disadvantages, which are the total amount of time and computation for the central authority to revoke a client, because it has to calculate the non-revocation proofs for all the pseudonyms revoked until that moment using digital signatures, and the latency of clients' authentications, because of the need of sending the RL to the client in the beginning of each authentication. The authors present two possible optimizations: usage of BFs for RLs decreasing their size and pre-computed non revocation proofs for pseudonyms already revoked to be inserted in future RLs. Even if the BFs are used to

implement RLs the problem of authentications' latency will still happen due to the size of BFs that are usually very big to reduce the amount of false positives.

3.1.3 Comparison of Pseudonym Schemes

We now summarize and discuss the main differences and accomplishments of existing pseudonym schemes presented in the previous chapters. Table 3.1 summarizes the properties offered by each scheme and the costs of the respective operations.

Analyzing Table 3.1 it is possible to notice that in all pseudonym schemes, the information required for storage in the pseudonym providers is dependent only on the number of clients registered. This is achieved by using pseudo random functions [8, 12], or by using a hash chain [7, 9], to generate pseudonyms, using seeds constructed from a user identifier. The fact that the pseudonyms issued to a client can be generated from the seed associated to that client, removes the need to store all the pseudonyms issued, as the pseudonym provider can generate them again, when revoking a client, and distribute them to the verifiers.

All the schemes based on epochs and time slots [7–9], have the number of pseudonyms issued dependent on the number of time slots and on the pseudonyms used per time slot. This situation is not ideal, because it forces the user to store many more pseudonyms than he will ever need. Also, this is problematic because of the number of pseudonyms that need to be revoked when a given client is revoked, increasing the size of RLs and the size of the "blacklists" of revoked pseudonyms in the verifiers.

It is also possible to notice the impossibility of schemes that are based on the notion of epochs and time slots [7–9], to provide perfect backward unlinkability. EDGAR improves this, by making it possible to reduce the size of the time slots, at the expense of having much bigger RLs. The growth of the size of the RLs is mitigated by the fact that this scheme allows to provide the clients only with as many pseudonyms as the number of their authentications, and by the latchkeys tree, as explained in Chapter 3.1.1.B, but it still does not achieve backward unlinkability perfectly. Although IFAL is a scheme based in epochs and time slots, it achieves the property by presenting a radical approach of issuing certificates to clients and activate them later by sending activation codes to the clients, every time slot, for the pseudonyms of that time slot. This scheme revokes clients by stop sending them the activation codes and waiting until the next time slot, for the client to run out of activating pseudonyms, which leads to the disadvantage of letting misbehaving users keep authenticating in the system until the end of the revocation time slot. This work has the advantage of swapping the distribution of revocation to the verifiers by the distribution of smaller activation codes to the users. This advantage is only an advantage in systems where the number of clients is similar to the number of verifiers. This does not happen in smart retail applications, where the number of customers is much higher than the number of verifiers. Despite of also achieving perfect backward unlinkability, V-token has a similar problem, it can't revoke a client right away after a

complaint, as it needs to let the revoked user run out of V-tokens.

Concerning the property of revocation auditability, it is observed that only Nymble, PEREA and Privacy Keeper provide it. Nymble and Privacy Keeper provide it by having the RL sent to the client in the beginning of each authentication. This RL is signed by the trusted central authority in order to prove its freshness and authenticity. This type of approach has the disadvantage of having high latencies in each authentication, due to the need of sending the RL to the user.

Only PEREA and Privacy Keeper achieve perfect backward unlinkability and revocation auditability simultaneously. PEREA achieves this using ZKPs and the scheme is impractical for physical systems, where users' authentications cannot be limited, such as smart retail applications. In this type of applications, users usually authenticate using a smart card or their smartphones, that have limited computational power and are not suitable for generating resource intensive ZKPs. Privacy Keeper achieves the properties solely, by asymmetric encryption, but does not achieve them efficiently. It has the disadvantages of high latencies in authentications and the high quantity of computation needed for the central authority to revoke a client. Although some anonymous authentication systems already provide the necessary properties, they do so in a highly inefficient manner, especially those that implement the property of revocation auditability, which requires transferring the RL to the client; a list that can grow to considerable sizes.

The substantial latency introduced by this process makes these systems unsuitable for many real-world applications, including smart retail environments.

Pseudonym Systems	Provider Storage	Pseudonyms Issued	Revocation List Size	**Backward Unlinkability**	**Revocation Auditability**
Haas *et al.* [7]	*O(N)*	*O(p*T)*	*O(R)*	Limited	No
EDGAR [8]	*O(N)*	*O(A)*	*O(R*A*log(T))*	Limited	No
Nymble [9]	*O(N)*	*O(p*T)*	*O(R)*	Limited	Yes
IFAL [10]	*O(N)*	*O(p*T)*	-	Full	No
V-token [1]	*O(N)*	*O(A)*	-	Full	No
PEREA [11]	-	*O(A)*	-	Full	Yes
Privacy Keeper [12]	*O(N)*	*O(A)*	*O(E*A)*	Full	Yes

Table 3.1: Comparison of the different systems. (N) Number of clients registered. (p) Pseudonyms per time slot. (T) Number of time slots in an epoch. (R) Number of revoked clients. (A) Number of authentications performed by a client. (E) Clients revoked since the beginning of the epoch.

3.2 Techniques for Optimizing Bloom Filter Transfer Efficiency

As mentioned before, some existing pseudonym systems implement the RLs as a BF, where revoked pseudonyms are stored. The RL is transferred multiple times, either for distributing the RL to all the verifiers, or during the authentication process, in systems that provide the property of **Revocation Auditability**, where the verifiers send the RL to the clients, so they can check their revocation status before

authenticating in the system.

BFs can grow to massive sizes, reaching several gigabytes, especially in highly dynamic systems, making their transfer really painful, particularly in the authentication process, making this process reach several seconds, making these systems absolutely infeasible and impossible of being applied to a real scenario. Therefore, we investigate two proposed techniques to improve the efficiency of the transfer process and evaluate their suitability for use in this context.

3.2.1 Compressed Bloom Filters

Compressed Bloom Filters [13] are a variation of the classic BFs designed to optimize their size for transmission. This work introduces a novel perspective on BFs, proposing that they can serve a dual purpose. Traditionally, a BF is treated as a data structure, optimized to minimize the probability of false positives given constraints like memory size and the number of items. However, when a BF is also utilized as a message for transmission, another key metric emerges: transmission size.

Reducing transmission size is critical in scenarios where minimizing network traffic is a priority, even if sufficient memory is available at the endpoint machines. Compression techniques can effectively decrease the number of bits transmitted, leading to improved efficiency. This work demonstrates how applying compression to BFs can enhance performance by reducing the volume of data sent over the network.

Nevertheless, this approach comes with certain trade-offs, particularly the additional computational overhead required for compression and decompression. The algorithm arithmetic coding [14] is recommended as a suitable solution in this context, as it offers a straightforward and efficient compression method tailored to such application.

This work proposes optimizing BF parameters with a focus not on minimizing the false positive rate but on achieving a specific probability of bits being set to 1, so they can be effectively compressed later. This approach aims to minimize the transmission size, enabling more efficient data transfer.

We consider the use of this technique to lower authentication latencies, in systems that support the **revocation auditability** property, considering verifiers to compressed BFs, sending less information on the network, and clients decompressing them to verify their revocation status.

To evaluate the applicability of this approach in the context of anonymous authentication using pseudonyms, we conducted a series of tests to measure the time required to decompress a BF. These tests varied the size of the original BF, that was compressed, being decompressed.

As illustrated in Figure 3.7 the decompression time increases significantly as the size of the compressed BF grows. In an authentication process, the client must receive the RL, the compressed BF, decompressing it each time authentication occurs. This substantial increase in decompression time renders this solution impractical for such use cases.

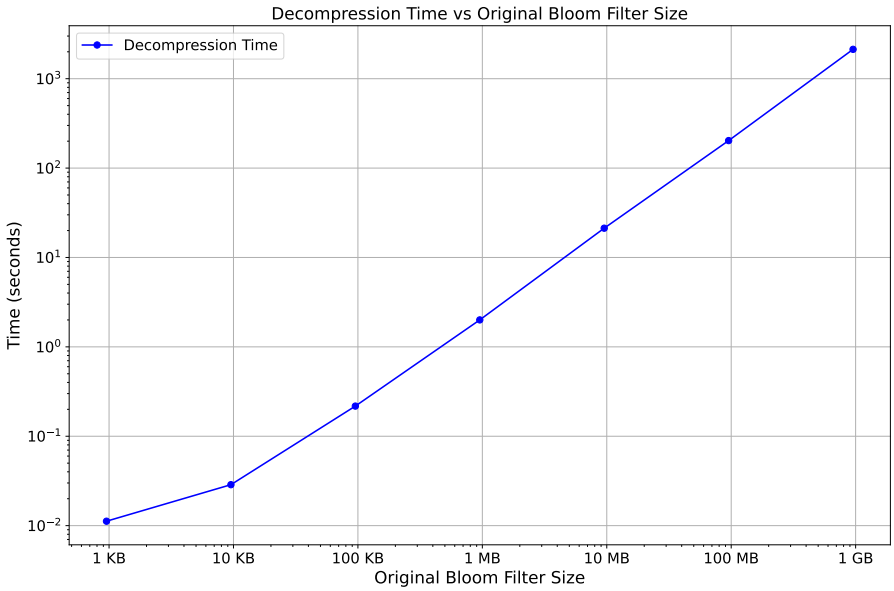


Figure 3.7: Compressed Bloom Filters Decompression

3.2.2 Compacted Bloom Filters

Compacted Bloom Filters [15] are another alternative design to the traditional BFs, aimed at optimizing their memory representation for storage and operational efficiency.

Compacted BFs need less memory than a BF to achieve the same false positive rate. Thus, when it is used in a network and distributed application, it reduces the transmission cost and the memory capacity needed at the endpoints. In contrast to Compressed BFs, the idea of using compression/decompression techniques is abandoned and it is proposed a new pattern to condense the bit vector.

For the creation of a Compacted BF, the original BF is divided in n multiple blocks, of b bits each. After the original BF is divided, it is translated as an array of n indices, each with m bits. Index i corresponds to the i^{th} value of $block_1$, $block_2$, ..., $block_n$. For example the first index of the Compacted BFs, has information pertaining the first bit of each $block$ of the splitted original BF.

The rules for populating each index of the compacted BF (`CmBFV`) are based on the bit patterns in the standard BF. Let S_i represent the set of i^{th} bits across all blocks in the BF. The process for determining the value of `CmBFV`[i] is as follows:

1. **All bits in S_i are "0":** If no "1" exists in S_i, the i-th bit in `CmBF` is set to 0:

$$\texttt{CmBFV}[i] = 0.$$

2. **Exactly one "1" in S_i:** If S_i contains exactly one "1", located in block r, we assign:

$$\texttt{CmBFV}[i] = r.$$

3. **All bits in S_i are "1":** If every bit in S_i is "1", the value is set to the maximum index value:

$$\texttt{CmBFV}[i] = 2^m - 1.$$

4. **More than half of the bits in S_i are "1":** If at least half of the bits in S_i are "1" (but not all), the value is also set to:

$$\texttt{CmBFV}[i] = 2^m - 1.$$

   This may cause some bits originally set to "0" to be interpreted as "1", leading to an increased false positive rate.

5. **Less than half of the bits in S_i are "1":** In this case, one block containing a "1" in the i^{th} position is randomly selected (e.g., block s), and we assign:

$$\texttt{CmBFV}[i] = s.$$

   This results in all other blocks containing "1" at the i^{th} position being interpreted as "0", which introduces the existence of false negatives.

To mitigate false positives and false negatives, all 2^m possible index values can be utilized to represent more unique bit patterns. This approach minimizes the number of bits misinterpreted as "1" or "0", reducing the overall error rates.

While this work provides an efficient method for compressing a BF, it introduces the occurrence of false negatives. This compromises its suitability for implementing RLs in authentication systems, as false negatives would enable revoked users to authenticate successfully.

3.2.3 Comparison of Bloom Filter Optimizating Techniques

Compressed Bloom Filters and Compacted Bloom Filters both reduce the size of traditional Bloom Filters but with different methods and trade-offs. Compressed BFs use compression techniques to minimize transmission size, which is beneficial in bandwidth-limited environments. However, the decompression process introduces computational overhead, making them less suitable for low-latency applications like authentication, where performance can be significantly impacted as the filter size grows.

In contrast, Compacted BFs optimize memory usage by reorganizing the Bloom Filter's bit array, dividing it in blocks and compressing them into smaller blocks. However, Compacted BFs introduce the possibility of false negatives, which can be problematic in systems like RL verification, where it is vital to accurately reject revoked users.

The above stated problems, make neither of the presented techniques suitable for this use case.

Summary

In this chapter we have presented the most relevant authentication schemes based on pseudonyms. Several of these schemes use Bloom Filter to encode lists of pseudonyms. We have also discussed a number of techniques that can be used to reduce the size of the filters exchanged during authentication.

4

Proposed Techniques

Contents

A disadvantage of pseudonym systems that provide the property of revocation auditability is that they require the transfer of the RL to the client, which, given its potentially large size for very dynamic systems, can introduce a significant latency in the process of authentication. Our work focuses on this problem, proposing techniques that aim to reduce the amount of information that clients and verifiers exchange during each authentication process, while maintaining all the security properties inherent to the RL, namely its integrity and authenticity. Considering that most of the pseudonym systems use BFs to implement the RL, we propose two techniques to achieve this: 1) a technique based on HBFAs, and 2) a technique inspired by Redactable Signatures. In both techniques proposed in this work, we assume the same entities as in most systems discussed in Chapter 3.1: clients, verifiers, and the central authority, as well as the same phases: pseudonym issuance, pseudonym revocation and access control, with substantial changes in the following phases: pseudonym revocation and access control.

4.1 Technique based on Hierarchical Bloom Filter Arrays

In this technique, we propose that the RL be composed of multiple BFs of different sizes, in addition to the original filter, each containing exactly the same elements. These filters are organized sequentially and in ascending order of their size.

4.1.1 Pseudonym Revocation

During the revocation of a client, the certification authority creates a set of n new BFs FBn_{η} to form the new RL. All the elements used to revoke the pseudonyms to be inserted in that list, being the elements the pseudonyms themselves or non-revocation proofs constructed for each revoked pseudonym, are inserted into all the BFs created. Finally, the central authority signs each of these filters with its private key, protecting the authenticity and integrity of each filter.

Subsequently, these filters and their respective digital signatures are aggregated into a RL, $RL_{new} = \langle [BF1, ..., BFn], [BF1^{K^{-}_{CA}}, ..., BFn^{K^{-}_{CA}}] \rangle$, which is sent to the verifiers.

4.1.2 Access Control

During the authentication process, represented in Figure 4.1, the client starts by downloading the first BF from the RL, which is the one with the smallest size, as well as, other relevant information for the pseudonym system, such as the random parameter η associated with the RL in Privacy Keeper. Then, they choose a pseudonym p and checks if this pseudonym is marked as revoked in the previously received filter. Since the false positive rate of a BF varies with its size, and the client begins by receiving the smallest filter, it is not unlikely that the client encounters a false positive at this step. In that case,

the client will then download the remaining filters from the RL in increasing order of size until they reach a filter where the pseudonym is not marked as revoked. If they reach the last filter and the pseudonym is marked as revoked in that filter, the client assumes that it has indeed been revoked and cancels the authentication. The authenticity of each filter is verified by the client using the certification authority's signature. If the client concludes that they have not been revoked, they can then complete the authentication by sending the pseudonym p and the corresponding proof to the verifier.

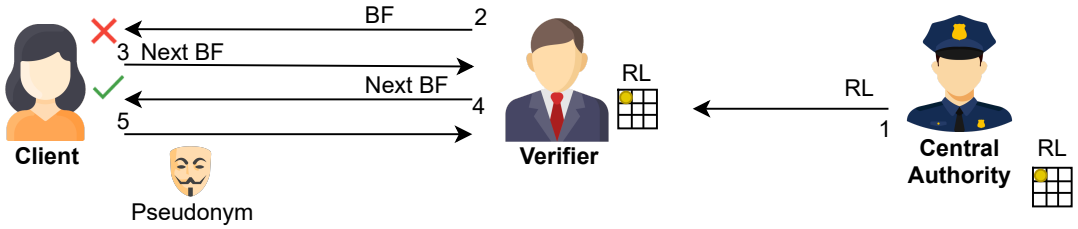


Figure 4.1: Authentication with Hierarchical Bloom Filter Arrays. BF is Bloom Filter. RL is Revocation List.

This technique is advantageous in cases where the client can confirm that the pseudonym has not been revoked by checking the first few filters (ideally, relying only on the smallest filter in most cases); however, it may worsen the original solution if the client has to download several filters. In most cases, the client would be able to authenticate using the first pseudonyms by checking only the smaller filters, making this an efficient solution overall.

4.2 Technique based on Redactable Signatures

In this technique, we take advantage of the fact that the client can know in advance the characteristics of the **Bloom filter** encoding the RL, such as its size, the number of hash functions. Using this information, the client can calculate their non-revocation proof and determine which entries in the vector they need to check to see if a given pseudonym has been revoked. We then propose that the client, instead of downloading the entire filter, only downloads segments of the filter in order to obtain all the necessary entries from the vector.

4.2.1 Pseudonym Revocation

By transferring only parts of the RL, the client is unable to verify the integrity of the list using a standard digital signature, as proposed in Privacy Keeper. To overcome this issue, we suggest that the central entity signs the RL using a technique inspired by redactable signatures.

In our solution, after creating the RL, $RL_{new} = \langle BF \rangle$, the BF is divided into multiple segments of configurable size, and the cryptographic hash of each segment is associated with the child nodes of a

MT. The remaining nodes of the tree are then generated recursively by hashing the concatenation of the hashes of the child nodes. Finally, the central authority generates a digital signature σ, by concatenating the root hash with the parameter η associated with the RL (unique value in Privacy Keeper or time slot in Nymble), as illustrated in Figure 4.2. The generated elements form the new RL, $RLnew = \langle BF, \sigma \rangle$, which is distributed to the verifiers, who then reconstruct the MT using the same process.

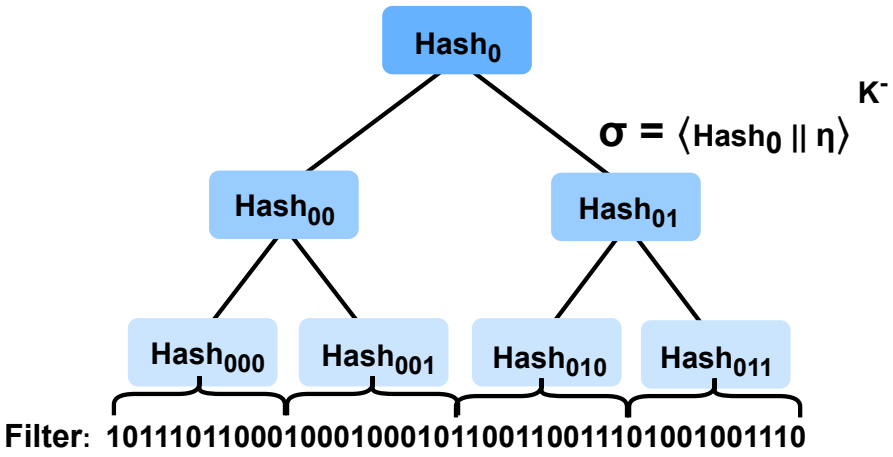


Figure 4.2: Redactable Signature Creation.

4.2.2 Access Control

In the access control phase, shown in Figure 4.3, the verifier begins by sending the information about the RL, such as: the characteristics of the BF, or the unique value η in the case of the anonymous system Privacy Keeper.

With this information, the client selects a pseudonym p that they have not used yet and with that pseudonym p, calculates the exact positions of the BF that needs to verify to test their revocation status, (in the case of the system Privacy Keeper, the client would first need to calculate the non-revocation proof), and requests these positions. The verifier responds by sending the filter segments that have the bits requested along with the necessary hashes for the client to reconstruct the path from the leaf nodes associated with the requested segments to the root node.

Upon receiving this information, the client ensures they are not revoked using the BF segments they received, checks the signature σ from the certifying entity, and then calculates the root hash recursively using the information sent by the verifier. The integrity of the BF segments received by the client is safe-guarded by the structure of the binary tree. If a malicious verifier were to modify any part, that change would reflect in some higher node of the tree due to the properties of the MT. The alteration would

necessarily affect the root hash, allowing the client to notice the change when verifying the signature from the certifying authority at that same node. If the client concludes that they are not revoked, they complete the authentication process by sending their pseudonym p and the corresponding proof to the verifier.

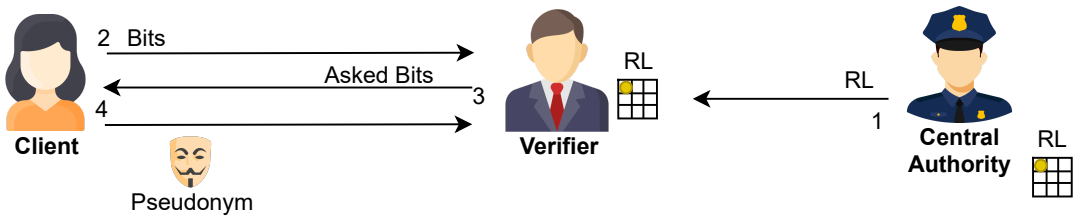


Figure 4.3: Authentication with Redactable Signatures. RL is Revocation List.

We propose an optimization to this technique specifically for cases where the client attempting to authenticate is not revoked, further reducing the amount of information propagated through the network. For non-revoked clients, it is sufficient to ensure that at least one bit of the bits needed to verify their pseudonym's validity, is set to zero in the BF. Accordingly, we introduce this optimization, having the verifier send only a single segment of the BF containing a bit requested by the client, that is set to ”0”, rather than transmitting all segments corresponding to the client's requested bits.

It is important to note that this optimization does not apply to revoked clients. Revoked clients must receive all the bits required to validate their pseudonym and confirm that each bit is set to ”1”, thereby verifying their revoked status.

Summary

In this chapter we have presented our proposed techniques to overcome the efficiency problem inherent to anonymous authentication protocols that transfer revocation lists. In the next chapter, we provide an extensive evaluation of these techniques, applying them to two already discussed anonymous authentication systems and comparing them between each other.

5

Evaluation

Contents

This chapter presents an experimental evaluation of the techniques proposed in Chapter 4. We star by describing some experiments that provide insights on the performance of each of the proposed techniques. Then we describe some experiments that allow to assess the relative performance of these techniques when compared with each other.

5.1 Experimental Setup

We have implemented both techniques in the Privacy Keeper system [12] and the Nymble system [9], programming each technique in C++. We developed a client and a verifier using two Intel NUC10i7FNB machines. The machines are equipped with an Intel i7-10710U processor, 16 GB of RAM, and Ubuntu 22.04 LTS. We used the Ed25519 algorithm [16] to generate deterministic digital signatures.

We analyze the impact of varying different parameters in both solutions on the amount of information transmitted during authentication. Furthermore, we evaluated both techniques by measuring the average latency of the authentication process while varying the size of the RL, comparing the results with the average latency of the original Privacy Keeper.

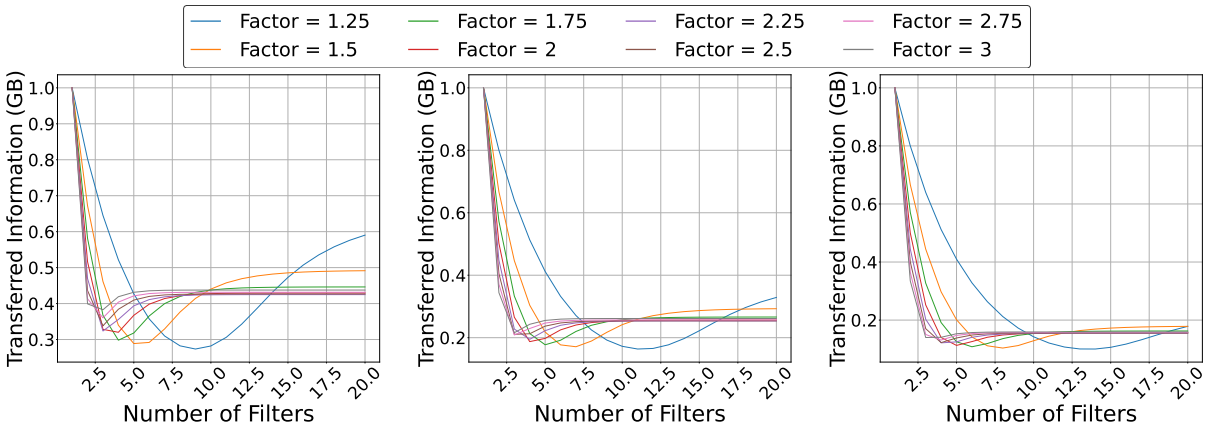
Furthermore, we assessed the effectiveness of both techniques by implementing them in the Privacy Keeper and Nymble systems and measuring the amount of information transferred during authentication across varying system scales and numbers of revoked clients. These results were then compared with the data transfer metrics in the original systems, providing an evaluation of the suitability of these techniques for real-world scenarios on different scales and across different anonymous authentication systems.

5.2 Individual Performance

5.2.1 Technique based on Hierarchical Bloom Filter Arrays

As previously described, our technique based on HBFA uses multiple filters of different sizes, with several parameters that directly influence the amount of information to be transferred during authentication. Thus, we evaluated the following parameters: 1) reduction factor between each filter, starting from the largest/original filter down to the smallest one. 2) the number of filters with different sizes used in the hierarchy, that is, how many times we reduce the largest filter by the chosen factor. 3) The false positive rate that we accept in the largest filter of the hierarchy. In Figure 5.1, we vary these parameters and calculate the expected amount of information related to BFs that needs to be transferred during authentication for each of these configurations, keeping the size of the largest filter fixed at 1GB.

It is worth noting that calculating the expected amount of transferred information is not trivial, as it



((a)) False Positive Rate = 0.1% **((b))** False Positive Rate = 0.01% **((c))** False Positive Rate = 0.001%

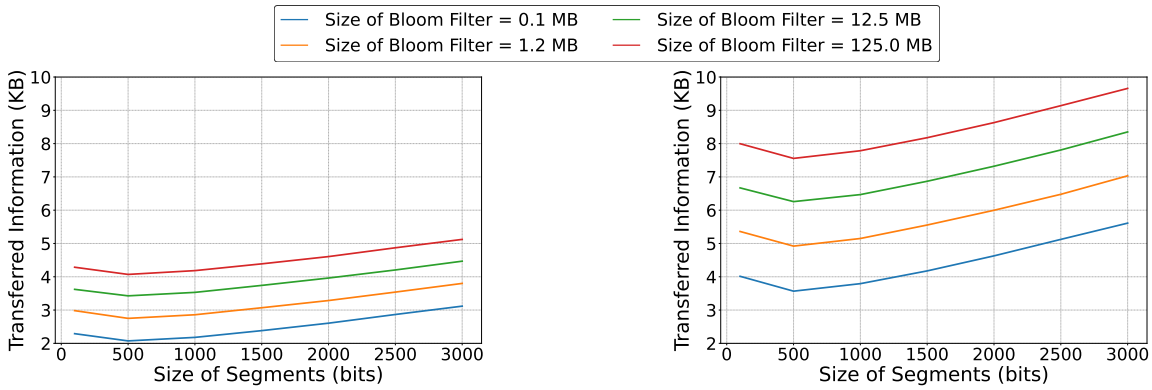
Figure 5.1: Variation of Parameters using Hierarchical Bloom Filters Arrays

depends on the number of filters the client needs to transfer before authenticating. To evaluate each set of parameters, we calculated the amount of information transferred for each scenario (in each scenario, the client needs to download a different number of filters before completing the audit process), and we calculated the expected value of the amount of information transferred by the client using the false positive rates of each BF to compute the probability of each scenario.

It can be observed that, when there is only one filter, the information required to transfer is always 1GB, representing the original filter. However, when we create more filters, the average amount of information transmitted quickly decreases to less than half with more than 5 filters. Furthermore, we observed that the smaller the reduction factor, the lower the expected amount of information transmitted during each authentication. However, it is necessary to increase the number of filters used to reach the minimum amount of information transmitted.

We also observed that the transferred information decreases when we reduce the false positive rate. This is due to the fact that reducing the rate also affects the smaller filters, as all filters will have lower false positive rates and a greater number of clients will authenticate using the first filters. Note that, in order to reduce the false positive rate while keeping the filter size fixed, it is necessary to reduce the number of items in the filter.

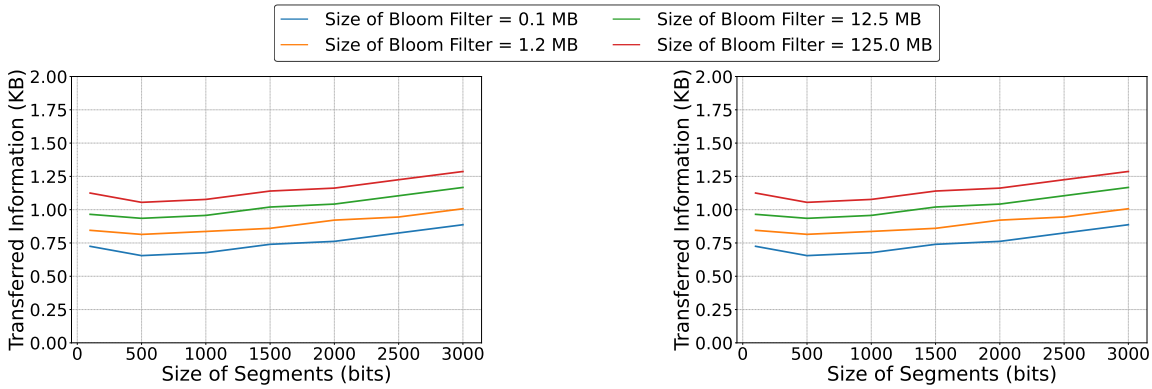
A good configuration for a false positive rate of 0.001% would be to choose a factor of 2 with 4 filters, as this is a point that minimizes the information to be transmitted over the network while maintaining a reasonable number of filters. Our technique drastically reduces the amount of information transferred, in particular, transferring only about 10% of the original amount of information on average.



((a)) Number of hash functions = 5

((b)) Number of hash functions = 10

Figure 5.2: Variation of Parameters using Redactable Signatures



((a)) Number of hash functions = 5

((b)) Number of hash functions = 10

Figure 5.3: Variation of Parameters using Redactable Signatures using proposed Optimization

5.2.2 Technique based on Redactable Signatures

Our technique based on RSigs has two main structures: the BF and a MT, which is constructed from the filter. The information transferred during authentication consists mainly of segments of the BF and various hashes from the MT.

Several parameters of the solution impact the amount of information transferred during authentication. These parameters include: 1) the number of hash functions, which directly influences the number of filter segments sent to the client, and 2) the size of the segments into which the filter is divided, since the segment size determines the number of filter segments that will be associated with the Merkle tree's leaf nodes, affecting the size of the MT and the number of hashes sent to the client.

In Figure 5.2, we varied these parameters and measured the average amount of information that the verifier sends to the client during authentication.

It is noticeable that increasing the number of hash functions in the filter increases the amount of information transmitted over the network. This is because the client receives more segments as the

number of hash functions increases and it also requires more hashes to compute the paths from the leaf nodes to the root node recursively.

The amount of information transferred during authentication reaches its minimum, considering the filter sizes and the number of hash functions studied, when the segment size is 500 binary digits. It becomes clear that with shorter segments, the associated MT will be larger and more hashes will be sent to the client. However, with larger segments, more information regarding these segments will be sent to the client. The 500 binary digits represent the optimal value between these two parameters.

From this experiment, we conclude that a good configuration would be to divide the BF into segments of 500 binary digits. In the worst-case scenario, if the filter had 10 hash functions, the solution would reduce the amount of information sent to the client from 125MB to 7KB. It should be noted that this represents a significant reduction in the original amount of information (specifically, 0.006% of the original size).

In order to evaluate the optimization proposed in Chapter 4.2.2, where non-revoked clients check their revocation status by checking only 1 bit set to ”0”, we set up a series of tests. In Figure 5.3 we varied the parameters exactly as in the Figure 5.2, to understand how much this optimization would improve the original technique based on RSigs.

It is possible to notice that with this optimization, the increase in the number of hash functions used does not increase the amount of information transferred during the authentication, this being a big advantage of this optimization. It allows one to increase the number of hash functions of the BF, thus, reducing the false positive rate, without increasing the amount of information transferred.

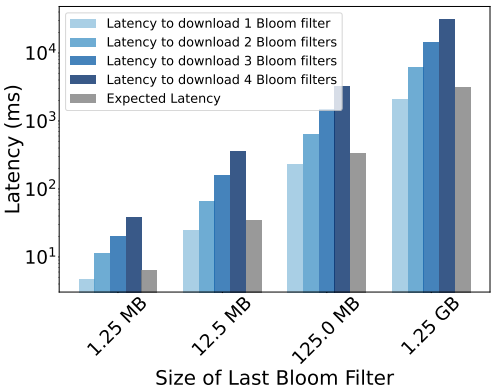
From this second experiment, we can conclude that the optimal configuration would still be to divide the BF into segments of 500 binary digits. With the proposed optimization, the solution would reduce the amount of information sent to the client from 125MB to 1KB, a reduction in the original amount of information (specifically 0. 0009% of the original size).

5.3 Comparative Performance

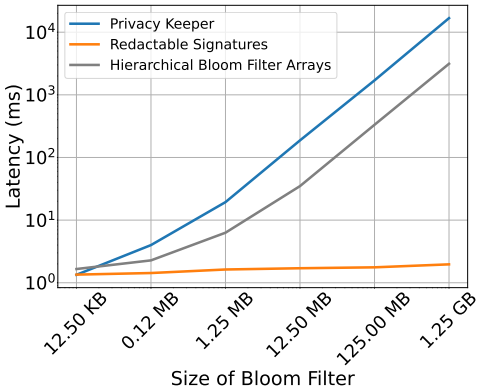
5.3.1 Evaluation Based on Revocation List Size

In this chapter, we evaluate the authentication latency of both proposed techniques implemented in Privacy Keeper and compare them with the authentication latency of the original Privacy Keeper. To ensure a fair comparison between the techniques, we chose configurations that minimize the amount of information exchanged during authentication between verifiers and clients, based on the analysis presented earlier.

For the evaluation of all systems, we developed a set of tests in which we measured the average authentication latency of a client, varying the size of the BF that implements the RL. In the case of the



((a)) Latency using Hierarchical Bloom Filter Arrays



((b)) Average Latency

Figure 5.4: Authentication Latency

HBFAs technique, this size corresponds to the size of the last and largest filter. For the evaluation of the technique with HBFAs, we used the following configuration: a reduction factor of 2, which means that each filter is half the size of the next filter, with the number of filters in the set fixed at 4, and 5 hash functions in each filter. By varying the latency of the last filter, we measured the authentication latency in each of the 4 possible scenarios (in each scenario, the client downloads a different number of filters). With these 4 latencies, we calculated the expected latency for each authentication, considering the latency for each of the 4 scenarios and the false positive rate of each filter, as shown in Figure 5.4(a). For the expected latency calculation, we considered a false positive rate of 0.01% for the last BF, which we used to calculate the false positive rates for the other filters. Note that only this technique has its efficiency dependent on the acceptable false positive rate in the RL. We used the expected latency obtained for comparison with the other two techniques. For the evaluation of the RSigs-based technique, we fixed the segment size at 500 binary digits and the number of hash functions at 5, measuring the authentication latency.

In Figure 5.4(b), we compare the expected latencies (and their evolution with the increasing size of the BF) using the two proposed techniques and the original Privacy Keeper (which always transfers the entire RL). Our experimental evaluation shows that, for BFs of 1.25GB, the authentication latency is, on average, $\pm$17 seconds with the original Privacy Keeper, $\pm$3 seconds with hierarchical Bloom filters, and $\pm$2 milliseconds with Redactable Signatures.

Therefore, the technique with RSigs demonstrates the best performance, significantly reducing the authentication latency by approximately 99.99% compared to the original Privacy Keeper.

5.3.2 Evaluation Based on System Scale

In this chapter, we evaluate the effectiveness of both proposed techniques implementing them in Privacy Keeper and Nymble, by measure the amount of information the verifier needs to send to the client during

the process of authentication, varying the number of revoked users in the systems, and comparing the techniques with the original systems. To ensure a fair comparison between the techniques, we selected configurations that minimize the information exchanged during authentication between verifiers and clients, as outlined in the earlier analysis.

As explained in Chapter 3.1, the number of revoked users affects the size of the revocation lists differently. In Privacy Keeper, a non-revocation proof for each revoked pseudonym is added to the revocation list. Consequently, the number of elements in the revocation list is equal to the number of revoked users multiplied by the average number of pseudonyms per revoked user. In Nymble, which operates using epochs and slots where each user is assigned a pseudonym for each time slot, revocation lists can be generated at the start of each slot independently. As a result, the number of elements in each revocation list corresponds exactly to the number of revoked users.

For both techniques, we used exactly the same configuration as in the evaluation of Chapter 5.3.1. For the evaluation of the technique with HBFAs, we used the following configuration: a reduction factor of 2, which means that each filter is half the size of the next filter, with the number of filters in the set fixed at 4, and 5 hash functions in each filter. For the evaluation of the RSigs-based technique, we fixed the segment size at 500 binary digits and the number of hash functions at 5. We consider a false positive rate of 0.01% for revocation lists. We consider, as well, 300 pseudonyms per revoked user, in Privacy Keeper.

We calculated the amount of transferred information for the HBFAs technique using the same approach described in Chapter 5.3.1. Specifically, we measured the information transferred in each of the four possible scenarios, where the client downloads a varying number of filters in each case. Using these four values, we determined the expected amount of transferred information per authentication by accounting for the information transferred in each scenario and the false positive rate of each filter, being 0.01% the rate of the last filter.

In Figure 5.5, we compare both techniques with the original systems. Our study shows, that both our proposed techniques are effective for both systems, allowing the systems to scale up without losing performance during the authentication protocol, being absolutely critical for the Privacy Keeper and its applicability and a huge improvement for Nymble. In particular, in a system with 1 million of revoked users, our technique based on RSigs, allows to lower the amount of transferred information from 1.09GB to 1.2KB, in the case of Privacy Keeper, and from 3.6MB to 0.9KB, in the case of Nymble.

5.4 Additional Time to Generate a RL

Both of our proposed techniques introduce additional time to the revocation list generation process. To evaluate the practicality of these techniques, we conducted a series of experiments aimed at assessing

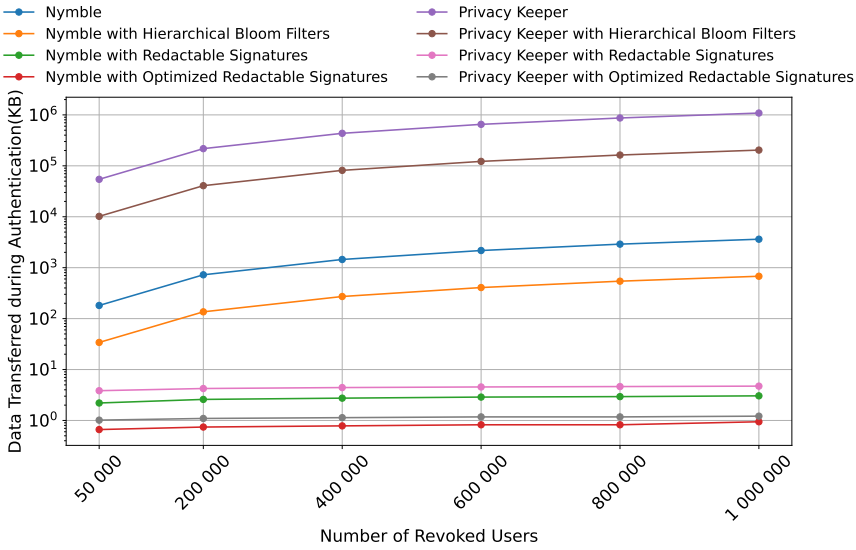
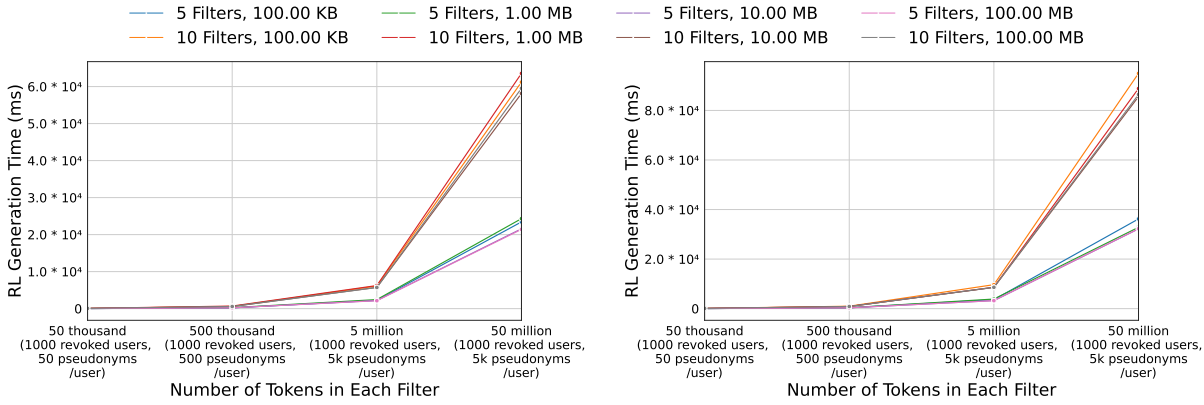


Figure 5.5: Data Transferred during Authentication vs Number of Revoked Users

their impact on generation time. The goal was to determine whether the overhead introduced remains within acceptable limits—ensuring that the delay does not become prohibitively high or hinder the timely revocation of misbehaving clients. These tests help us understand the trade-offs involved and confirm the feasibility of implementing the proposed approaches in real-world scenarios.

5.4.0.A Technique based on Hierarchical Bloom Filter Arrays



(a) 5 hash functions

(b) 10 hash functions

Figure 5.6: HBFA's Overhead

Figure 5.6 illustrates the results of a series of tests we conducted to assess the additional overhead involved in generating the revocation list using the Hierarchical Bloom Filter Arrays technique, as well as

its impact on the latency required to generate a revocation with this approach.

There are several factors that may impact the necessary additional overhead in generating a revocation list with multiple Bloom filters, instead of just one. We have conducted a series of tests to measure the additional latency this would take, varying parameters such as: the number of filters that our revocation list was composed of, the number of hash functions of each filter, the size of those filters and the number of tokens to be inserted in each filter.

By analyzing our results it is possible to notice that the number of hashes, the number of tokens to be inserted and the number of filters, have a linear impact in the additional overheard, doubling the latency when doubling the number of hashes or the number of tokens, due to the number of bits that are necessary to set to ”1” also double. We can also conclude that the size of the Bloom filters do not have a significant impact on this metric, due to the nature of Bloom filters being highly efficient and constant with respect to their operations.

Also, we can conclude that, depending on the system scale, the additional overhead can be negligible, being only a few milliseconds for most cases until 5 million tokens, starting to impose a serious overhead when we hit the scale of 50 million tokens, reaching the magnitude of 60 seconds or more.

5.4.0.B Technique based on Redactable Signatures

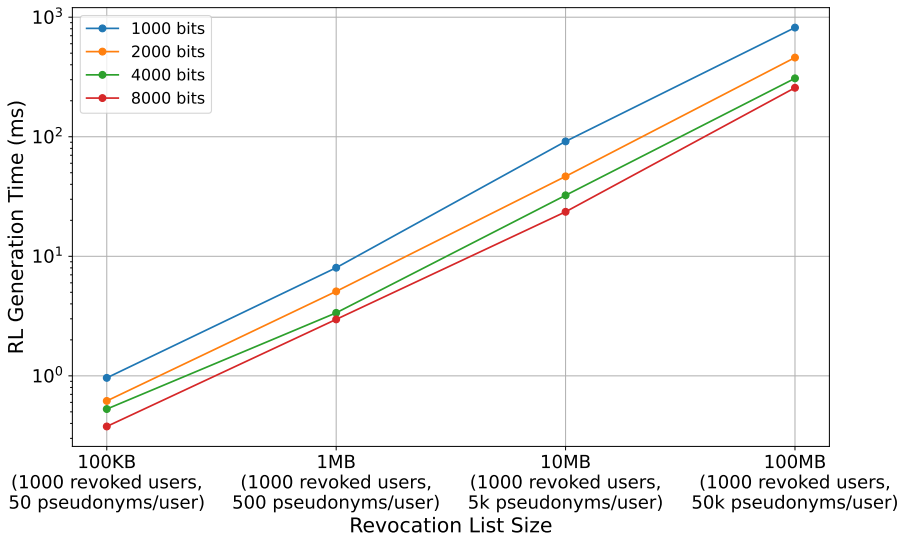


Figure 5.7: Redactable Signature Overhead.

To quantify the overhead introduced by our Redactable Signature technique, we conducted experiments measuring the time required to generate a signature over a revocation list, varying both the size of the Bloom filter and the size of the chunks into which it was divided.

As shown in Figure 5.7, the signature generation time increases linearly with the Bloom filter size,

as expected. Additionally, using smaller chunks results in longer generation times due to the larger number of leaf nodes in the Merkle tree, which increases the number of required hash computations. In contrast, larger chunks yield smaller trees and thus lower computational overhead. The latency required to generate a redactable signature ranges from just 1 millisecond or less for Bloom filters of 100KB to hundreds of milliseconds for filters as large as 100MB.

This highlights a trade-off between redaction granularity and efficiency. As discussed before, (see Figures 5.2 and 5.3), larger chunks reduce signature generation time but require more data transfer during authentication. This trade-off becomes particularly relevant for Bloom filters exceeding 100MB. For smaller filters, generation time remains below one second across all chunk sizes, making the overhead negligible in practice.

Summary

In this chapter we have presented the experimental evaluation of our work. We run several tests, implementing both techniques on different anonymous authentication systems, comparing the efficacy of both techniques between each other and between the original systems, showing that both techniques can be applied to this type of systems improving their efficiency, being the techniques absolutely crucial for the adoption of these systems in the future. In the next chapter, we conclude our work and discuss the limitations of our work and what can be done for future work, to further improve anonymous authentication.

6

Conclusion

Contents

In this chapter we present some conclusions.

6.1 Conclusions

Applications such as smart retail have seen an enormous growth in recent years, with the increasingly popular use of technologies in retail, in particular digital technologies. While these technologies can be used to improve customer experience, they also enable smart retail providers to profile user behaviour, not preserving customer privacy. Pseudonyms can be a solution for this problem, allowing customers to authenticate in these smart retail systems using identities other than their real identities.

Several pseudonym schemes have been proposed for several applications, but most of them do not achieve all the properties that we consider relevant for smart retail applications, such as perfect **backward unlinkability** and **revocation auditability**. The systems that achieve them, do so at the expense of high latencies in the authentication procedure, making these systems impossible to be applied in a real case scenario.

In this work, we introduce two distinct techniques designed to minimize the amount of information exchanged during the authentication process in anonymous authentication systems, thereby reducing authentication latency, keeping all the desired security and privacy properties.

We experimentally demonstrate that one of the techniques significantly reduces the authentication time from several tens of seconds to just a few milliseconds. This makes anonymous authentication systems feasible in the smart retail domain and in other areas of interest where privacy concerns are raised.

Bibliography

[1] F. Schaub, F. Kargl, Z. Ma, and M. Weber, "V-tokens for conditional pseudonymity in vanets," in *2010 IEEE Wireless Communication and Networking Conference*. IEEE, 2010, pp. 1–6.

[2] J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," in *Advances in Cryptology—CRYPTO 2002: 22nd Annual International Cryptology Conference Santa Barbara, California, USA, August 18–22, 2002 Proceedings 22*. Springer, 2002, pp. 61–76.

[3] J. Li, N. Li, and R. Xue, "Universal accumulators with efficient nonmembership proofs," in *Applied Cryptography and Network Security: 5th International Conference, ACNS 2007, Zhuhai, China, June 5-8, 2007. Proceedings 5*. Springer, 2007, pp. 253–269.

[4] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[5] Y. Zhu, H. Jiang, and J. Wang, "Hierarchical Bloom filter Arrays (HBA): a novel, scalable metadata management system for large cluster-based storage," in *Proceedings of the IEEE International Conference on Cluster Computing*, San Diego (CA), USA, 2004, pp. 165–174.

[6] R. Johnson, D. Molnar, D. Song, and D. Wagner, "Homomorphic signature schemes," in *Cryptographers' track at the RSA conference*. Springer, 2002, pp. 244–262.

[7] J. J. Haas, Y.-C. Hu, and K. P. Laberteaux, "Efficient certificate revocation list organization and distribution," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 3, pp. 595–604, 2011.

[8] C. Correia, M. Correia, and L. Rodrigues, "Using range-revocable pseudonyms to provide backward unlinkability in the edge," in *Proceedings of the ACM Conference on Computer and Communications Security. Copenhagen, Denmark*, 2023.

[9] P. P. Tsang, A. Kapadia, C. Cornelius, and S. W. Smith, “Nymble: Blocking misbehaving users in anonymizing networks,” *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 2, pp. 256–269, 2009.

[10] E. Verheul, C. Hicks, and F. D. Garcia, “Ifal: Issue first activate later certificates for v2x,” in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2019, pp. 279–293.

[11] P. P. Tsang, M. H. Au, A. Kapadia, and S. W. Smith, “Perea: Towards practical ttp-free revocation in anonymous authentication,” in *Proceedings of the 15th ACM conference on Computer and communications security*, 2008, pp. 333–344.

[12] C. Correia, “Low-latency privacy-preserving access to edge storage,” Ph.D. dissertation, Instituto Superior Tecnico, Universidade de Lisboa, Jul. 2024.

[13] M. Mitzenmacher, “Compressed bloom filters,” in *Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*, 2001, pp. 144–150.

[14] A. Moffat, R. M. Neal, and I. H. Witten, “Arithmetic coding revisited,” *ACM Transactions on Information Systems (TOIS)*, vol. 16, no. 3, pp. 256–294, 1998.

[15] N. Mosharraf, A. P. Jayasumana, and I. Ray, “Compacted bloom filter,” in *2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC)*, 2016, pp. 304–311.

[16] D. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, “High-speed high-security signatures,” *Journal of cryptographic engineering*, vol. 2, no. 2, pp. 77–89, 2012.